世界知的所有権機関
国際事務局

PCT

特許協力条約に基づいて公開された国際出願

(51) 国際特許分類6
G06F 9/38, 15/82

A1

(11) 国際公開番号 WO99/27439

(43) 国際公開日 1999年6月3日(03.06.99)

(21) 国際出願番号 PCT/JP98/05230

(22) 国際出願日 1998年11月19日(19.11.98)

(30) 優先権データ
特願平9/362473 1997年11月20日(20.11.97) JP
特願平10/282118 1998年10月5日(05.10.98) JP

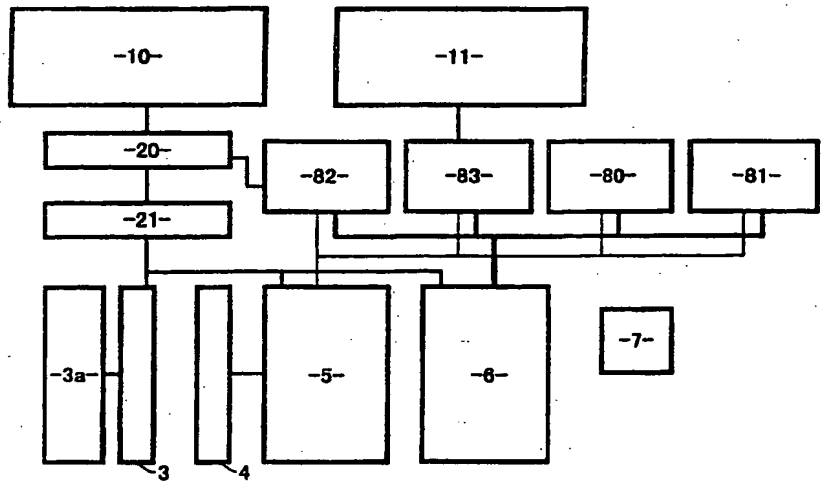
(71) 出願人；および
(72) 発明者
関 一(SEKI, Hajime)[JP/JP]
〒790-0848 愛媛県松山市道後喜多町4番38号 Ehime, (JP)

(81) 指定国 AU, BG, BR, CA, CN, CU, CZ, HU, ID, IL, IS, JP, KR, MX, NO, NZ, PL, SG, SK, TR, UA, US, VN, ARIPO特許 (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), ユーラシア特許 (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), 欧州特許 (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI特許 (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

添付公開書類
国際調査報告書

(54)Title: COMPUTER SYSTEM

(54)発明の名称 計算機システム



(57) Abstract

A computer system for processing a program described in the machine language of a stack machine, comprising a data cache (11), an integrated register file (6) in which data is written in each of entries, an advanced pointer stack (3) and a completion pointer stack (4) in which the addresses of the entries of the integrated register file are written into entries, an instruction queue (5) having the construction of an FIFO in which the contents of each of instructions are written into each entry, arithmetric units (80 and 81), and a loading/storing unit (83). When the instruction held in the first entry of the instruction queue can be completed or has been completed, the completion pointer stack is operated so as to execute the same operation of the advanced pointer stack when the held instruction is decoded on the basis of the contents of the first entry of the instruction queue, and the first entry is eliminated from the instruction queue.

(57)要約

データ・キャッシュ（１１）と各々のエントリにデータが書き込まれるようになっている統合レジスタ・ファイル（６）と、各々のエントリに統合レジスタ・ファイルのエントリのアドレスが書き込まれるようになっている前進ポインタ・スタック（３）及び完了ポインタ・スタック（４）と、各々のエントリに個々の命令の内容が書き込まれるようになっているＦＩＦＯの構成となっている命令キュー（５）と、演算ユニット（８０），（８１）と、ロード／ストア・ユニット（８３）を具備する、スタックマシンの機械語で記述されたプログラムを処理する計算機システムである。

命令キューの先頭のエントリにおいて保持されている命令の完了が可能である、あるいはそうなると、その命令キューの先頭のエントリの内容に基づき、保持されている命令がデコードされた際の前進ポインタ・スタックの動作を再現すべく完了ポインタ・スタックを操作し、命令キューからその先頭のエントリを除外するようになっている。

明　細　書

計算機システム

技術分野

本発明は、スタックマシンの機械語で記述されたプログラムを高速で処理する新規な構成の計算機システムに関するものである。

背景技術

従来、スタックマシンにおいては、命令の実行は、基本的にプログラム上の順序通り（ in-order ）に行われるものであった。すなわち、スタックマシンにおける演算命令は、オペランド・スタックからソース・データをポップし、演算を実行し、その演算結果をオペランド・スタックにプッシュするというような動作を指示するものであるが、このような命令の連鎖として書かれたプログラムを逐次的に実行するのである。

このような従来のスタックマシンにおいては、命令をプログラム上の順序通り（ in-order ）に実行するので、制御構造が単純なもので済むという利点があるが、処理速度が制約を受けるという問題点があった。

そこで、スタックマシンの機械語で記述されたプログラムを out-of-order で処理するような計算機方式が考案された。例えば、日本特公平２－２６００８２号、米国特許第５５２２０５１号や、米国特許第５３３３３２０号及び米国特許第５７６５０１４号におけるプロセッサ要素がある。これらの明細書に示されるプロセッサは、処理性能の向上という点で十分ではない上に、正確な例外処理を保証する上で問題があった。

本発明は、上記問題点を解決するため創案されたものであり、正確な

例外処理を保証しつつ、スタックマシンの機械語で記述されたプログラムを out-of-order でより効率的に処理する計算機システムを提供することを目的としている。

発明の開示

本発明による計算機システムは、データ・キャッシュと、各々のエントリにデータが書き込まれるようになっている統合レジスタ・ファイルと、各々のエントリに統合レジスタ・ファイルのエントリのアドレスが書き込まれるようになっているスタックの構成となっている前進ポインタ・スタック及び完了ポインタ・スタックと、各々のエントリに個々の命令の内容が書き込まれるようになっているＦＩＦＯキューの構成となっている命令キューと、演算を実行するようになっている演算ユニットと、データ・キャッシュ及び統合レジスタ・ファイルにアクセスできるようになっているロード／ストア・ユニットとを具備する。

従来のスタックマシンにおいて、スタックが ....., word1, word2, word3, word4（右端がスタックトップ）となっている状態は、本発明による計算機システムにおいて、ポインタ・スタックが .....,〈a〉,〈b〉,〈c〉,〈d〉（右端がスタックトップ）で、エントリ・アドレスが〈a〉,〈b〉,〈c〉及び〈d〉である統合レジスタ・ファイルの各エントリに、それぞれ word1, word2, word3 及び word4 が保持されている状態に対応する。

本発明の計算機システムにおいては、命令がデコードされるごとに、その命令の内容に応じて前進ポインタ・スタック及び統合レジスタ・ファイルを操作すると共にその命令の内容を命令キューに書き込むようになっている。この際、命令に含まれるオペランド・スタックに対するスタック操作が、前進ポインタ・スタックに対して同様に適用される。ここ

で、１語のデータのオペランド･スタックへのプッシュ操作を、本発明の計算機システムにおいてエミュレートするには、割り付けられていない統合レジスタ･ファイルの１エントリをそのデータを保持すべく割り付け、そのエントリのアドレスを前進ポインタ･スタックにプッシュすればよい。

即ち、オペランド･スタックに対するポップ操作を含む命令がデコードされた場合には、ポップすべき語数と同じ数だけ統合レジスタ･ファイルのエントリのアドレスを前進ポインタ･スタックからポップする。オペランド･スタックに対するプッシュ操作を含む命令がデコードされた場合には、プッシュすべき語数と同じ数だけ割り付けられていない統合レジスタ･ファイルのエントリを割り付け、上記割り付けた統合レジスタ･ファイルのエントリのアドレスを前進ポインタ･スタックにプッシュする。さらに、デコードされた命令の内容を、ポップ／プッシュ操作を伴う命令の場合にはポップ／プッシュされる統合レジスタ･ファイルのエントリのアドレスと共に、命令キューに書き込むようになっている。

命令キューに保持されている未実行の命令は、データ駆動（各々の動作が、必要なすべてのソース･データが揃い実行可能となった時点で実行されること）の原理に基づき out-of-order で処理されるようになっている。

例えば、命令キューにおいて、演算命令を書き込みの内容とし、必要なソース･データが全て統合レジスタ･ファイルに書き込み済みとなっているエントリがあり、演算ユニットが利用できる状態であれば、その演算の実行を開始する。演算の実行が正常に終了すれば、演算結果をデスティネーションである統合レジスタ･ファイルのエントリに書き込む。

命令キューの先頭のエントリにおいて保持されている命令の完了が可能である、あるいはそうなると、その命令キューの先頭のエントリの内

容に基づき、保持されている命令がデコードされた際の前進ポインタ・スタックの動作を再現すべく完了ポインタ・スタックを操作し、命令キューからその先頭のエントリを除外し、ポップ操作によって完了ポインタ・スタックにおけるアドレスの保持が無くなった統合レジスタ・ファイルのエントリの割り付けを解除するようになっている。

図面の簡単な説明

第１図は、本発明にかかる好ましい計算機システムの基本構成を示すブロック図、第２図は、前進ポインタ・スタック及び完了ポインタ・スタックの構成を示す図、第３図は、後述する本発明第１実施例における統合レジスタ・ファイルの各々のエントリの詳細な構成を示す図、第４図は、命令キューの構成を示す図、第５図は、命令キューの各々のエントリの詳細な構成を示す図、第６図～第１２図は、本発明第１実施例における一動作例の、サイクル毎の前進ポインタ・スタック、完了ポインタ・スタック、命令キュー及び統合レジスタ・ファイルの内容を具体的に示した説明図、第１３図は、本発明第１実施例において、１サイクル当り３命令までデコードできるような構成をとる場合に、プログラムがどのように変換されるかを具体的に示す図表、第１４図は、後述する本発明第２実施例における統合レジスタ・ファイルの各々のエントリの詳細な構成を示す図、第１５図～第２１図は、本発明第２実施例における一動作例の、サイクル毎の前進ポインタ・スタック、完了ポインタ・スタック、命令キュー及び統合レジスタ・ファイルの内容を具体的に示した説明図である。

発明を実施するための最良の形態

以下に、本発明にかかる好ましい計算機システムについて、図面を参

照しながら説明する。なお、以下に述べる本発明による計算機システムの実施例は、Java Virtual Machine（Java VM）で規定されるスタックマシンの基本的な命令をハードウエアで実行するものである。すなわち、データ語長を 32 ビットとして、これを単位にロード／ストア及び算術論理演算等の演算を行う。従って、例えば、倍長語の間での算術演算は、２語づつ合せて４語のソース・データをもとに２語の演算結果を生ずる。

従来のスタックマシンにおける、語の単位でデータがプッシュ／ポップされるようになっているスタックは、後述するポインタ・スタックと区別するために、以降では、ワード・スタックと呼ぶことにする。

Java VM にはもともとハードウェアで実行することを想定していない複雑な命令が含まれるが、以下に述べる本発明による計算機システムの実施例は、次のような基本的な命令をハードウェアで実行するものとする。

（ａ）即値データのオペランド・スタックへのプッシュ命令

bipush, sipush, aconst_null, iconst_m1, iconst_<i>, fconst_<f>, lconst_<l>, dconst_<d>

（ｂ）変数データのオペランド・スタックへのロード命令

ldc1, ldc2, iload, iload_<n>, fload, fload_<n>, aload, aload_<n>, ldc2w, lload, lload_<n>, dload, dload_<n>, iaload, laload, faload, daload, aaload, baload, caload, saload

（ｃ）オペランド・スタック上のデータの変数へのストア命令

istore, istore_<n>, fstore, fstore_<n>, astore, astore_<n>, lstore, lstore_<n>, dstore, dstore_<n>, iastore, lastore, fastore, dastore, aastore, bastore, castore, sastore

（ｄ）演算命令

（ｄ－１）算術演算命令

iadd, ladd, fadd, dadd, isub, lsub, fsub, dsub, imul, lmul, fmul, dmul, idiv, ldiv, fdiv, ddiv, irem, lrem, frem, drem, ineg, lneg, fneg, dneg

（ｄ－２）論理演算命令

ishl, ishr, iushr, lshl, lshr, lushr, iand, land, ior, lor, ixor, lxor

（ｄ－３）変換演算命令

i2l, i2f, i2d, l2i, l2f, l2d, f2i, f2l, f2d, d2i, d2l, d2f, int2byte, int2char, int2short

（ｄ－４）比較演算命令

lcmp, fcmpl, fcmpg, dcmpl, dcmpg

（ｅ）オペランド･スタックの操作命令

pop, pop2, dup, dup2, dup_x1, dup2_x1, dup_x2, dup2_x2, swap

（ｆ）分岐命令

ifeq, ifnull, iflt, ifle, ifne, ifnonnull, ifgt, ifge, if_icmpeq, if_icmpne, if_icmplt, if_icmpgt, if_icmple, if_icmpge, goto, goto_w

以降、特にことわらない限り、「命令」とは上に挙げた命令のいずれかを意味するものとする。

以下に、オペランド･スタックの操作命令の処理方法の異なる、第１及び第２の２つの実施例について説明する。

まず、本発明第１実施例の計算機システムについて説明する。

第１図は計算機システムのブロック図であって、１０は命令キャッシュ、１１はデータ･キャッシュ、２０は命令フェッチ･ユニット、２１は命令デコード･設定ユニット、３は前進ポインタ･スタック、３ａは前進ポインタ･スタック履歴ファイル、４は完了ポインタ･スタック、５は命令キュー、６は統合レジスタ･ファイル、７はフリー･リスト、８０及び８１は各々演算ユニット０及び１、８２は分岐ユニット、８３はロード／ストア･ユニットを表している。

次に、本発明第１実施例の計算機システムの各構成要素ごとにその詳

細な構成を説明する。

（A）命令フェッチ･ユニット

命令フェッチ･ユニットは、図示してないプログラムカウンタ（pc レジスタ）を具備しており、命令キャッシュから命令をフェッチし、命令デコード･設定ユニットに渡す。分岐の予測や分岐の実行も担う。

（B）命令デコード･設定ユニット

命令デコード･設定ユニットは、命令フェッチ･ユニットから渡された命令のデコードを行い、命令に含まれる演算等がデータ駆動で実行されるように、後述する前進ポインタ･スタック、命令キュー及び統合レジスタ･ファイル等を設定するための各種信号を発生する。

（C）ポインタ･スタック

ポインタ･スタックは、各々のエントリに統合レジスタ･ファイルのエントリのアドレスが書き込まれるようになっているスタックの構成となっている。

従来のスタックマシンにおいて、ワード･スタックが ....., word1, word2, word3, word4（右端がスタックトップ）となっている状態は、本発明による計算機システムにおいて、ポインタ･スタックが .....,〈a〉,〈b〉,〈c〉,〈d〉（右端がスタックトップ）で、エントリ･アドレスが〈a〉,〈b〉,〈c〉及び〈d〉である統合レジスタ･ファイルの各エントリに、それぞれ word1, word2, word3 及び word4 が保持されている状態に対応する。

本発明の計算機システムは、前進ポインタ･スタック（ＡＰＳ；Advanced Pointer Stack）と完了ポインタ･スタック（ＣＰＳ；Completed Pointer Stack）の２つのポインタ･スタックを具備する。

本発明の計算機システムにおいては、個々の命令がデコードされるごとに、前進ポインタ･スタック（以下ではＡＰＳで示す）及び統合レジ

スタ・ファイルを操作すると共に命令の内容を命令キューに書き込むことにより、命令に含まれる演算等がデータ駆動で実行されるべく設定される。すなわち、前進ポインタ・スタックはデコード・設定済みの全ての命令によるスタック操作を反映している。

他方、完了ポインタ・スタック（以下ではＣＰＳで示す）は、プログラム上の順番で完了済みの全ての命令によるスタック操作を反映するものである。本発明の計算機システムはデータ駆動の原理に基づくout-of-order 実行を可能とするものであるが、完了ポインタ・スタックは、正確な例外処理を保証するため、プログラムが in-order で実行された場合の状態を構成するために存在するものである。

ポインタ・スタック及び統合レジスタ・ファイルのエントリ数は限られたものであるので、ワード・スタックが成長すると、ポインタ・スタック及び統合レジスタ・ファイルを用いてスタック・トップ近傍の部分しか保持できない。本実施例の計算機システムにおいては、ワード・スタックの残りの部分はデータ・キャッシュに格納されるようになっている。そのため、各ポインタ・スタックは循環型のバッファの構成となっており、プッシュ・ポインタとボトム・ポインタと呼ぶ２つのレジスタが各々存在する。プッシュ・ポインタは、統合レジスタ・ファイルのエントリのアドレスを保持する最上位のエントリの１つ上を示す。ボトム・ポインタは、統合レジスタ・ファイルのエントリのアドレスを保持する最下位のエントリを示す。ボトム・ポインタの値からプッシュ・ポインタの値を引くことで、ポインタ・スタックに何エントリの空きがあるかがわかる。初期状態においては、プッシュ・ポインタ及びボトム・ポインタの各々の値は共に０となっている。

第２図は、本実施例の計算機システムにおける各ポインタ・スタックと各プッシュ・ポインタ及びボトム・ポインタの関係を示す説明図であ

る。２つのポインタ・スタックＡＰＳ３及びＣＰＳ４は同数のエントリを有し、各ポインタ・スタックで各々のエントリに下から順に０、１、２、・・・とアドレスが付けられているものとする。縦線が施されているエントリは統合レジスタ・ファイルのエントリのアドレスを保持しているものとする。第２図に示すように、プッシュ・ポインタは、ＡＰＳ及びＣＰＳの各々に対して設けられており、それぞれ PP_OF_APS 及び PP_OF_CPS と名付けている。他方、ボトム・ポインタは１つだけ設けられており、これがＡＰＳ及びＣＰＳで共用される。これを BP_OF_PS と名付けている。

ＡＰＳとＣＰＳの間には、エントリの数だけ比較回路が設けられており、ＡＰＳ及びＣＰＳの同じエントリ・アドレスにある（第２図において水平に並ぶ）エントリの間でその内容が比較されるようになっている。

命令に含まれるオペランド・スタックに対する１語分のプッシュ操作に対応して割り付けられる統合レジスタ・ファイルの１エントリのアドレスをＡＰＳの PP_OF_APS で示されるエントリに書き込み、PP_OF_APS の値に１を加えるようになっている。逆に、命令に含まれるオペランド・スタックに対する１語分のポップ操作に対応して、PP_OF_APS の値から１を引くようになっている。ＣＰＳと PP_OF_CPS に関しても同様である。

BP_OF_PS で示されるエントリの内容がＡＰＳとＣＰＳで一致する場合には、その２つのポインタ・スタックで一致する内容で示される統合レジスタ・ファイルのエントリに書き込まれている１語分のデータをデータ・キャッシュにストア（Spill）することができる。その際、BP_OF_PS の値に１を加えるようになっている。逆に、データ・キャッシュにストア（Spill）したデータを統合レジスタ・ファイルにロード（Fill）する場合には、最後にストア（Spill）した１語分のデータに対し、フリー・リ

ストに登録されている統合レジスタ・ファイルの１エントリを割り付けてそのデータを書き込み、その統合レジスタ・ファイルのエントリのアドレスをＡＰＳ及びＣＰＳの BP_OF_PS で示されるエントリの１つ下に各々書き込み、BP_OF_PS の値から１を引くようになっている。

本実施例の計算機システムは、分岐予測に基づく投機的実行を可能にするために、前進ポインタ・スタック履歴ファイル（以下では「ＡＰＳ履歴ファイル」と記す）を具備する。ＡＰＳ履歴ファイルの各々のエントリには、ＡＰＳの全エントリ及び PP_OF_APS の内容が書き込めるようになっている。

（D）統合レジスタ・ファイル（ＣＲＦ；Consolidated Resister File）

統合レジスタ・ファイル（以下ではＣＲＦで示す）は、従来のスタックマシンにおけるワード・スタックの内容を、順序不同で保持するものである。

第３図は、本第１実施例における、ＣＲＦ６の各々のエントリ６(i)の詳細な構成を示す説明図である。ここで、ｉはエントリのアドレスである。ＣＲＦ６の各々のエントリ６(i)はデータ・フィールド６１(i)、書込み完了フラグ（ＷＣＦ，Write Completion Flag）フィールド６２(i)、カラー（Ｃ，Colour）フィールド６３(i)及びビジービット（ＢＢ）フィールド６４(i)から成っている。

実際のＣＲＦのハードウェア上の構成は、上述の各フィールド別に設けられたレジスタ・ファイルの集合体である。

ＣＲＦの各々のエントリのデータ・フィールドは、１語分のデータが書き込まれる構成となっている。

ＣＲＦの各々のエントリにおいて、ＷＣＦフィールドは、データ・フィールドにデータの書き込みが完了していれば１、完了していなければ０が書き込まれているようになっている。

ＣＲＦの各々のエントリにおいて、Ｃフィールドは、そのＣＲＦのエントリが、命令に含まれるプッシュ操作に対応して割り付けられたものであるのか、アンダーフロー回避のためのデータ・キャッシュからのロード(Fill)の際に割り付けられたものであるのかの区別、前者の場合にはさらに分岐タグが書き込まれるようになっている。本実施例においては、後述するように、分岐タグはＡＰＳ履歴ファイルのエントリのアドレスと一定の関係にある。

ＣＲＦの各々のエントリにおいて、ＢＢフィールドは、そのＣＲＦのエントリがデータを保持すべく割り付けられている状態であれば１、割り付けられていない状態であれば０が書き込まれているようになっている。

（Ｅ）フリー・リスト（ＦＬ）

フリー・リスト（以下ではＦＬで示す）は、フリーな、即ち、割り付けられていない（ＢＢフィールドが０である）ＣＲＦのエントリのアドレスを保持するためのメモリであり、本実施例においては、循環型のＦＩＦＯキューの構成となっている。

初期状態においては、ＣＲＦの全てのエントリのアドレスがＦＬに登録されている。ＣＲＦのフリーなエントリを割り付ける必要がある場合に、ＦＬからフリーなＣＲＦのエントリのアドレスが取り出される。逆に、ＣＲＦのあるエントリの割り付けが解除されれば、そのエントリのアドレスがＦＬに登録されるようになっている。

（Ｆ）命令キュー（ＩＱ；Instruction Queue）

命令キュー（以下ではＩＱで示す）は、実行または完了を待っているデコード・設定済の命令を保持するメモリであり、循環型のＦＩＦＯキューの構成となっている。

第４図は、ＩＱの構成を示す説明図である。第４図において、ＩＱ５

の各々のエントリは下から順に０、１、２、・・・とアドレスが付けられているものとし、縦線が施されているＩＱ５のエントリは、実行または完了を待っている命令を保持しているものとする。ＩＱは、設定ポインタ／完了ポインタと名付けた二つのレジスタを具備する。設定ポインタは、次にデコード･設定される命令の内容を書き込むべきエントリを示す。完了ポインタは、次に完了されるべき命令のエントリを示す。完了ポインタの値から設定ポインタの値を引くことで、ＩＱに何エントリの空きがあるかがわかる。初期状態においては、設定ポインタ及び完了ポインタの値は共に０となっている。

第５図は、ＩＱ５の各々のエントリ５(i)の詳細な構成を示す説明図である。ここで、ｉはエントリのアドレスである。ＩＱ５の各々のエントリ５(i)はオペレーション･フィールド５００(i)、オペランド･フィールド５０１(i)、第１ソース･フィールド５１０(i)、第１書込み完了フラグ（ＷＣＦ１）フィールド５１１(i)、第２ソース･フィールド５２０(i)、第２書込み完了フラグ（ＷＣＦ２）フィールド５２１(i)、第３ソース･フィールド５３０(i)、第３書込み完了フラグ（ＷＣＦ３）フィールド５３１(i)、第４ソース･フィールド５４０(i)、第４書込み完了フラグ（ＷＣＦ４）フィールド５４１(i)、第１デスティネーション･フィールド５５(i)、第２デスティネーション･フィールド５６(i)、分岐タグ（ＢＴ）フィールド５７(i)、及び実行状態（Ｓ；State）フィールド５８(i)から成っている。

ＩＱの各々のエントリのオペレーション･フィールドはオペレーション･コードが書き込まれる構成となっている。

ＩＱの各々のエントリのオペランド･フィールドは、オペレーション･コードに続いてオペランドが示されるような命令の場合に、このオペランドが書き込まれるようになっている。

ＩＱの各々のエントリの第１～第４ソース・フィールドの各々は、ソース・データを保持すべく割り付けられているＣＲＦのエントリのアドレスが書き込まれるようになっている。オペランド・スタックに対するポップ操作を含む命令の場合には、命令によりポップされるべきデータを保持すべく割り付けられているＣＲＦのエントリのアドレスが、ポップされる順に書き込まれるようになっている。

ＩＱの各々のエントリの第１～第２デスティネーション・フィールドの各々は、命令のデコード・設定に伴い、新たに割り付けられるＣＲＦのエントリのアドレスが書き込まれるようになっている。オペランド・スタックに対するプッシュ操作を含む命令の場合には、命令によりプッシュされるべきデータを保持すべく割り付けられるＣＲＦのエントリのアドレスが、プッシュされる順に書き込まれるようになっている。

ＩＱの各々のエントリにおいて、第１～第４の各ＷＣＦフィールドは各々第１～第４ソース・フィールドに対応して設けられている。ＷＣＦ１フィールドは第１ソース・フィールドに示されるＣＲＦのエントリにデータの書き込みが完了していれば１、完了していなければ０が書き込まれているようになっている。第２～第４のＷＣＦフィールド、ソース・フィールドに関しても同様である。

ＩＱは、各エントリのソース・フィールドごとに比較回路を備えており、データの書き込みが行われるＣＲＦのエントリのアドレスを各ソース・フィールドの内容と比較して、一致するソース・フィールドに対応するＷＣＦフィールドに１を立てるような機能を有する。

ＩＱの各々のエントリのＢＴフィールドは、分岐予測に基づく投機的実行に係るもので、本実施例においては、後述するように、ＢＴフィールドに書き込まれる分岐タグはＡＰＳ履歴ファイルのエントリのアドレスと一定の関係にある。

ＩＱの各々のエントリにおいて、Ｓフィールドは、そのエントリに書き込まれている命令の実行状態に応じて、未実行、実行済み、正常終了、例外事象発生等の情報が書き込まれているようになっている。

（Ｇ）演算ユニット

本実施例の計算機システムは、演算ユニット０及び演算ユニット１を具備しており、その各々は、ＩＱより送られてくる算術論理演算、データ・タイプの変換演算、比較演算等の演算命令を実行する機能を有し、互いに独立に並行して動作することができるようになっている。

本発明の計算機システムにおいては、各々の演算ユニットをパイプライン化したり、より多くの演算ユニットを具備したり、演算ユニットごとに実行する演算の種類を特定した構成とすることも可能である。

（Ｈ）分岐ユニット

分岐ユニットは、ＩＱより送られてくる条件分岐命令を処理し、分岐の有無を確定して、命令フェッチ・ユニットに通知する機能を有する。

（Ｉ）ロード／ストア・ユニット（ＬＳＵ；Load/Store Unit）

ロード／ストア・ユニット（以下ではＬＳＵで示す）は、アドレス計算を行う機能を有し、データ・キャッシュ及びＣＲＦにアクセスすることができるようになっている。

ＬＳＵは、最初のローカル変数へのポインタを保持する図示してないレジスタ（ vars レジスタ）を具備する。本実施例の計算機システムにおいては、最初のローカル変数の格納域はデータ・キャッシュあるいはＣＲＦにあるが、vars レジスタには、データ・キャッシュにおける相当するアドレス値が書き込まれているようになっている。すなわち、全てあるいは一部のローカル変数の実際の格納域がＣＲＦにある場合でも、各々のローカル変数に、全てのローカル変数をデータ・キャッシュにストア（Spill）したと仮定した場合のデータ・キャッシュにおけるアドレ

ス値を対応させることができるので、ロード／ストア命令の処理において、ＬＳＵは vars レジスタの値を用いてアドレス計算を行い、対象となるローカル変数の格納域がデータ・キャッシュかＣＲＦかを判定し、その格納域にアクセスする。格納域がＣＲＦにあると判定された場合には、アクセスすべきＣＲＦのエントリのアドレスはＡＰＳから読み出される。

ロード／ストア命令が命令デコード・設定ユニットでデコードされると、その内容が、命令キューに書き込まれると同時に、ＬＳＵにも送られるようになっている。

ＬＳＵは、プログラム上の順番で、ロード／ストア命令を命令キューのエントリのアドレス、対象となる変数名及びデータと共に蓄える図示してないキューを具備する。このキューは連想機能を備えており、変数名を照合してデータ・アクセスの依存性の検証を行うことにより、ロード命令の実行を out-of-order で行うことができるようになっている。本実施例の計算機システムにおいては、ローカル変数の格納域はデータ・キャッシュあるいはＣＲＦにあるが、上記キューにすでに同じ変数名の書き込みがあるローカル変数のロード命令の場合、変数データは上記キューより読み出される。ストア命令の実行は、正確な例外処理を保証するために、後述するように、完了ステージにおいて in-order で行われる。

ＬＳＵは、プログラム中に示されるロード／ストア命令を実行すると共に、オーバーフロー／アンダーフローの回避のため、ＡＰＳ、ＣＰＳ及びＣＲＦの空きに応じて、ＣＲＦに保持されているスタックの最下位にあたるデータをデータ・キャッシュとの間で自動的にストア(Spill)／ロード(Fill)するようになっている。

本発明第１実施例の計算機システムにおいては、 ＡＰＳ、ＣＰＳ及びＣＲＦのオーバーフロー／アンダーフローは以下に示すようなメカニ

ズムにより回避される。

ＡＰＳ、ＣＰＳあるいはＣＲＦの空きが一定量以下になると、オーバーフローを回避するために以下のような制御動作が随時行われるようになっている。

すなわち、ＡＰＳがＣＲＦのエントリのアドレスを一定量以上保持していない場合、あるいは、BP_OF_PS で示されるエントリの内容がＡＰＳとＣＰＳで一致しない場合、以上の条件が解消するまでの間、命令シーケンスの流れを停止するよう命令フェッチ･ユニットに信号が送られる。

逆に、ＡＰＳがＣＲＦのエントリのアドレスを一定量以上保持しており、BP_OF_PS で示されるエントリの内容がＡＰＳとＣＰＳで一致する場合には、その２つのポインタ･スタックで一致する BP_OF_PS で示されるエントリの内容で示されるＣＲＦのエントリに書き込まれている１語分のデータをデータ･キャッシュにストア（Spill）し、BP_OF_PS の値に１を加える。さらに、上記ＣＲＦのエントリのＢＢフィールドを０に変更し、そのエントリのアドレスをＦＬに登録する。

ＡＰＳ、ＣＰＳ及びＣＲＦの空きがいずれも一定量以上になると、アンダーフローを回避するために、最後にストア（Spill）した１語分のデータをデータ･キャッシュから取り出し、それにＦＬに登録されているフリーなＣＲＦの１エントリを割り付け、そのデータ･フィールドに書き込む。ＷＣＦ、ＢＢの各フィールドは１とする。さらに、その割り付けられたＣＲＦのエントリのアドレスを、ＡＰＳ及びＣＰＳの BP_OF_PS で示されるエントリの１つ下に各々書き込み、BP_OF_PS の値から１を引く。

また、ＣＲＦとデータ･キャッシュの間の Spill/Fill の動作を効率的に行うために、ＬＳＵがデータをいったん蓄えるバッファを備え、この中

に適当な語数のデータを溜めておくような構成とすることも可能である。

ついで、本発明第１実施例の計算機システムの動作を説明する。

本実施例の計算機システムは命令を、①命令フェッチ、②命令デコード･設定、③実行、④完了の４ステージで処理する。当分の間、説明を簡単にするため、１サイクルで１つの命令をデコード･設定／完了できるものとして、以下に各ステージごとに動作内容を説明する。

① 命令フェッチ･ステージ

このステージでは、命令フェッチ･ユニットが命令キャッシュから命令を取り出すと共に、次にフェッチする命令のアドレスを決定する。次に命令をフェッチするのは通常次アドレス値からであるが、フェッチした命令が無条件分岐命令であるか、条件分岐命令で分岐すると予測した場合、分岐予測が外れた場合、あるいは例外が発生した場合には、フェッチするアドレス値を変更する。

② 命令デコード･設定ステージ

このステージでは、命令をデコードして、命令の内容に応じて前進ポインタ･スタック（ＡＰＳ）及び統合レジスタ･ファイル（ＣＲＦ）を操作すると共に命令の内容を命令キュー（ＩＱ）に書き込むことにより、命令に含まれる演算等がデータ駆動で実行されるべく設定する。以下に、設定動作を詳細に説明する。

本発明の計算機システムにおいては、従来のスタック･マシンにおけるワード･スタックのスタックトップ近傍がポインタ･スタックとＣＲＦによって再現されるが、命令に含まれるオペランド･スタックに対するスタック操作が、ＡＰＳに対して同様に適用される。

１語のデータのオペランド･スタックへのプッシュ操作をエミュレートするには、ＦＬに登録されているフリーなＣＲＦの１エントリをその

データを保持すべく割り付け、そのエントリのアドレスをＡＰＳにプッシュすればよい。

オペランド・スタックの操作命令（Java VM における pop, pop2, dup, dup2, dup_x1, dup2_x1, dup_x2, dup2_x2, swap）の場合、基本的には、オペランド・スタックに対して行うべき操作をＡＰＳに対して同様に行えばよい。本第１実施例においては、スタック上でコピーを作成するようなオペランド・スタックの操作命令（ Java VM における dup, dup2, dup_x1, dup2_x1, dup_x2, dup2_x2 ）の場合には、コピー・データを保持すべくＦＬに登録されているフリーなＣＲＦのエントリを割り付け、そのエントリのアドレスをＡＰＳの適切なエントリに書き込むようになっている。

命令のデコード・設定に伴い新たに割り付けられるＣＲＦのエントリにおいては、ＢＢフィールドに１を立て、Ｃフィールドには命令デコード・設定ユニットから送られてくる分岐タグを書き込む。即値データのプッシュ命令の場合には、データがすでに得られているので、データ・フィールドにそのデータを書き込み、ＷＣＦフィールドに１を立てる。それ以外の場合には、データはデコード・設定の時点では得られていないので、ＷＣＦフィールドを０としておく。

デコードされた命令の内容をプログラム上の順番でＩＱに保持しておくために、その命令の内容をＩＱの設定ポインタで示されるエントリに書き込み、設定ポインタの値に１を加える。すなわち、オペレーション・フィールドにオペレーション・コードを書き込み、オペレーション・コードに続いてオペランドが示されるような命令の場合には、オペランド・フィールドにこのオペランドを書き込む。ＢＴフィールドには命令デコード・設定ユニットから送られてくる分岐タグを書き込む。Ｓフィールドに関しては、無条件分岐命令、即値データのオペランド・スタック

へのプッシュ命令あるいはスタック上でコピーを作成することのないオペランド・スタックの操作命令（Java VM における pop, pop2, swap）の場合は実行済みとし、その他の命令の場合は未実行としておく。

オペランド・スタックに対するポップ操作を含む命令の場合には、ポップすべき語数と同じ数だけＡＰＳからポップされるＣＲＦのエントリのアドレスを、その順で第１～第４ソース・フィールドに書き込む。この際、エントリ・アドレスがポップされるＣＲＦのエントリの各々でＷＣＦフィールドを読み出し、ＩＱの対応するＷＣＦフィールドに送る。

オペランド・スタックに対するプッシュ操作を含む命令の場合には、プッシュすべき語数と同じ数だけＡＰＳにプッシュされるＣＲＦのエントリのアドレスを、その順で第１～第２デスティネーション・フィールドに書き込む。

本第１実施例においては、スタック上でコピーを作成するようなオペランド・スタックの操作命令の場合には、コピー元となるデータを保持すべく割り付けられているＣＲＦのエントリのアドレスをソース・フィールドに、コピー・データを保持すべく新たに割り付けられるＣＲＦのエントリのアドレスをデスティネーション・フィールドに、一定の対応関係のもとに書き込む。

命令の種類に応じて、オペランド・スタックに対してポップ／プッシュすべき語数（オペランド・スタックの操作命令の場合には、作成するコピーの語数）は決まっているので、オペレーション・フィールドの内容によって、第１～第４ソース・フィールド、第１～第４のＷＣＦフィールド及び第１～第２デスティネーション・フィールドのうちのいずれが有効であるかを知ることができる。

ロード／ストア命令の場合には、その内容を、ＩＱに書き込むと同時に、その書き込みが行われるＩＱのエントリのアドレスと共にＬＳＵに

送る。

③ 実行ステージ

ＩＱに保持されている未実行の命令は、データ駆動で処理される。従って、命令実行順序は out-of-order になる。以下に、命令の種類ごとに、実行ステージにおける動作を説明する。

（ａ）即値データのオペランド･スタックへのプッシュ命令

実行ステージにおける動作としては、何も行わない。

（ｂ）変数データのオペランド･スタックへのロード命令

ＩＱの、ロード命令を書き込みの内容とするエントリに関しては、同じ内容が命令デコード･設定ステージにおいてＬＳＵに送られている。ＬＳＵでは、送られてきたロード命令を out-of-order で処理するようになっている。

ＩＱにおいて、オペランド･スタック上のデータをポップしてアドレス計算を行うようなロード命令（Java VM における iaload, laload, faload, daload, aaload, baload, caload, saload）を書き込みの内容とし、有効なソース･フィールドに対応するＷＣＦフィールドが全て１となっている（ソース･データが全てＣＲＦに書き込み済みとなっている）エントリがあれば、そのエントリのアドレスと共にＣＲＦをアクセスして得られるソース･データをＬＳＵに送る。ＬＳＵは送られてきたソース･データをもとにアドレス計算を行い、ロードの実行を試みる。

ＬＳＵで変数データが得られると、デスティネーションであるＣＲＦのエントリのデータ･フィールドに変数データを書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドで上記デスティネーションであるＣＲＦのエントリのアドレスを照合し、一致するソース･フィールドに対応するＷＣＦフィールドを１とする。この際、同じタイミングで命令が書き込まれるＩＱのエントリにおいて

は、その書き込まれる内容と比較するようになっている。以上の動作が正常に終了すれば、そのロード命令を保持しているＩＱのエントリのＳフィールドを正常終了に変更する。

（ｃ）オペランド･スタック上のデータの変数へのストア命令

ＩＱにおいて、ストア命令を書き込みの内容とし、有効なソース･フィールドに対応するＷＣＦフィールドが全て１となっている（ソース･データが全てＣＲＦに書き込み済みとなっている）エントリがあれば、そのエントリのアドレスと共にＣＲＦをアクセスして得られるソース･データをＬＳＵに送る。

オペランド･スタック上のデータをポップしてアドレス計算を行うようなストア命令（Java VM における iastore, lastore, fastore, dastore, aastore, bastore, castore, sastore）の場合、ＬＳＵは送られてきたソース･データをもとにアドレス計算を行う。

以上の動作が正常に終了すれば、そのストア命令を保持しているＩＱのエントリのＳフィールドをストア実行可能に変更する。

正確な例外処理を保証するために、実際のストアの実行は完了ステージにおいて行う。

（ｄ）演算命令

ＩＱにおいて、演算命令を書き込みの内容とし、有効なソース･フィールドに対応するＷＣＦフィールドが全て１となっている（ソース･データが全てＣＲＦに書き込み済みとなっている）エントリがあり、利用可能な状態の演算ユニットがあれば、そのエントリの内容をそのエントリのアドレス及びＣＲＦをアクセスして得られるソース･データと共に利用可能な演算ユニットに送り実行させる。

演算の実行が正常に終了すれば、デスティネーションであるＣＲＦのエントリのデータ･フィールドに演算結果を書き込み、ＷＣＦフィール

ドを１に変更する。その上、ＩＱにおけるＣＲＦのエントリのアドレスの照合及びＷＣＦフィールドの変更を、上述のロード命令の場合と同様に行う。以上の動作が正常に終了すれば、その演算命令を保持しているＩＱのエントリのＳフィールドを正常終了に変更する。

（ｅ）オペランド･スタックの操作命令

スタック上でコピーを作成することのないオペランド･スタックの操作命令に関しては、実行ステージにおける動作としては、何も行わない。

本第１実施例においては、ＩＱにおいて、スタック上でコピーを作成するようなオペランド･スタックの操作命令を書き込みの内容とし、有効なソース･フィールドに対応するＷＣＦフィールドが全て１となっている（ソース･データが全てＣＲＦに書き込み済みとなっている）エントリがあれば、その有効なソース･フィールドに示されるＣＲＦのエントリからデータを読み出し、これを対応するデスティネーション･フィールドに示されるＣＲＦのエントリのデータ･フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおけるＣＲＦのエントリのアドレスの照合及びＷＣＦフィールドの変更を、上述のロード命令の場合と同様に行う。以上の動作が正常に終了すれば、そのオペランド･スタックの操作命令を保持しているＩＱのエントリのＳフィールドを正常終了に変更する。

（ｆ）分岐命令

無条件分岐命令に関しては、実行ステージにおける動作としては、何も行わない。

ＩＱにおいて、条件分岐命令を書き込みの内容とし、有効なソース･フィールドに対応するＷＣＦフィールドが全て１となっている（ソース･データが全てＣＲＦに書き込み済みとなっている）エントリがあれば、そのエントリの内容をそのエントリのアドレス及びＣＲＦをアクセスし

て得られるソース・データと共に分岐ユニットに送り実行させる。

分岐ユニットにおいて条件式の計算が正常に終了すれば、その結果を分岐先アドレスと共に命令フェッチ・ユニットに通知する。以上の動作が正常に終了すれば、その条件分岐命令を保持しているＩＱのエントリのＳフィールドを正常終了に変更する。

以上のように、ＩＱに保持されている未実行の命令は、実行可能となったものから処理されるので、命令実行順序は out-of-order になる。また、演算ユニット０，１、分岐ユニット及びロード／ストア・ユニットの各実行ユニットは互いに独立に並行して動作する。

ある命令の処理において例外事象が発生した場合には、その情報を、その命令を保持しているＩＱのエントリのＳフィールドに書き込むと共に、命令フェッチ・ユニットに例外ベクタを通知する。

④ 完了ステージ

ある命令が完了できるためには、プログラム上の順番でその命令よりも前にある命令が全て完了していなくてはならない。

ＩＱの完了ポインタで示されるエントリにおいて、Ｓフィールドが実行済み／正常終了である、あるいはそうなると、そのエントリに書き込まれている命令の内容に基づいてＣＰＳ及びＣＲＦを操作し、完了ポインタの値に１を加える。

ＣＰＳは、命令がデコード・設定された際のＡＰＳの動作を再現すべく操作される。すなわち、ポップ／プッシュ操作を含む命令の場合には、有効なソース・フィールドの内容と同じものを順にＣＰＳからポップし、有効なデスティネーション・フィールドの内容を順にＣＰＳにプッシュする。スタック上でコピーを作成することのないオペランド・スタックの操作命令の場合には、オペランド・スタックに対して行うべき操作をＣＰＳに対して全く同様に行えばよい。本第１実施例においては、スタ

ック上でコピーを作成するようなオペランド・スタックの操作命令の場合には、有効なソース・フィールド及び有効なデスティネーション・フィールドを参照して、その命令のデコード・設定の際にＡＰＳに対して行われた操作がＣＰＳにおいて再現される。

本第１実施例においては、上述のＣＰＳに対する操作に伴い、エントリ・アドレスがＣＰＳからポップされるＣＲＦのエントリでは、ＢＢフィールドを０に変更し、そのエントリ・アドレスをＦＬに登録する。

ＩＱの完了ポインタで示されるエントリにおいて、ストア命令が書き込まれている場合には、Ｓフィールドがストア実行可能である、あるいはそうなると、ＬＳＵに、上記ストア命令を保持しているＩＱのエントリのアドレスを示して、実際のストアの実行を依頼する。こうすれば、データが in-order でストアされることが保証できる。さらに、ＣＰＳ及びＣＲＦに対する操作を上と同様に行い、完了ポインタの値に１を加える。

以上のように、完了ポインタの値に１が加えられることによって、キューから除外されたＩＱのエントリに保持されていた命令は、完了したことになる。その命令よりも前にデコード・設定された命令はすべて完了しているので、命令の完了は in-order で行われることになる。

ＩＱの完了ポインタで示されるエントリにおいて、Ｓフィールドが例外事象発生である、あるいはそうなった場合には、その時点におけるＣＰＳ及びＣＲＦによって、プログラムが in-order で実行された場合の例外発生時点の状態が構成されるので、正確な例外処理が可能である。例外事象の発生した命令以降にデコード・設定された命令を全てキャンセルするには、キャンセルされるべき命令が書き込まれているＩＱのエントリの有効なデスティネーション・フィールドに示されるＣＲＦのエントリの各々に対して、そのＢＢフィールドを０に戻し、そのエントリ・

アドレスをＦＬに登録することによって、割り付けを解除し、完了ポインタの値に１を加えたものを設定ポインタに書き込むことによって、キャンセルされるべき命令を保持しているＩＱのエントリを全てキューから除外すればよい。

以上が、本発明第１実施例の計算機システムの動作についての全般的な説明であるが、ついで、具体的な動作例について説明する。

いま、本第１実施例の計算機システムで、以下のようなプログラムを実行することを考えよう。

dload [A]　(変数名[A]に対応する倍精度浮動小数点データのロード)
dload [B]　(変数名[B]に対応する倍精度浮動小数点データのロード)
dadd　(倍精度浮動小数点データ間の加算)
d2f　(倍精度浮動小数点データの単精度浮動小数点データへの変換)
fload [T]　(変数名[T]に対応する単精度浮動小数点データのロード)
dup_x1　(スタックトップの語のコピーを作成し、先頭から３語目に割り込ませる)
fdiv　(単精度浮動小数点データ間の除算)
fsub　(単精度浮動小数点データ間の減算)
fstore [X]　(スタックトップにある単精度浮動小数点データの変数名[X]に対応する格納域へのストア)

以上のプログラムは、X=T-(A+B)/T　の計算を行うものであるが、ＡとＢのデータが倍精度で与えられ、この間の加算を倍精度のまま実行して、得られた加算データを単精度に変換して、以降は単精度で計算を行う、というものである。

第６図～第１２図は、本第１実施例の計算機システムにおいて、上記プログラムを処理する際の動作をサイクル毎に示した説明図であり、以

下ではこの図をもとに詳細な動作を説明する。第６図～第１２図において、ＣＲＦ６及びＩＱ５の各エントリの構成は、それぞれ第３図、第５図のものと同じである。第６図～第１２図で空白となっている箇所は、そのフィールドの内容に留意する必要が無いことを意味する。時系列で各構成要素の内容を示すために、各部の符号の後尾にハイフンと各サイクルに対応する数字を添えている。また、第６図～第１２図において、ＡＰＳ、ＣＰＳ、ＩＱ及びＣＲＦの各エントリは下から順に０、１、２、～ のようにアドレスが付けられているものとする。

本動作例においては、説明を簡単にするため、変数データは全てデータ･キャッシュに格納され、ＣＲＦとデータ･キャッシュの間の Spill/Fill の動作は行わないものとする。従って、BP_OF_PS の値は終始０である。

また、本動作例においては、当初、ＡＰＳ、ＣＰＳ、ＩＱ及びＣＲＦは初期化されており、ＦＬにＣＲＦの全てのエントリのアドレスが順に〈０〉，〈１〉，〈２〉，〈３〉‥‥と書き込まれていて、この順で取り出されるものとする。

以下に、各サイクルにおける動作を、（Ａ）命令デコード･設定、（Ｂ）実行及び（Ｃ）完了の各ステージに分けて詳細に説明する。

（１－Ａ）第１サイクルの命令デコード･設定ステージ

命令 **dload** ［A］のデコード･設定を行う。倍長語の変数データのオペランド･スタックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦの２エントリ６(0)、６(1)をそのデータを保持すべく割り付け、そのエントリのアドレス〈０〉，〈１〉をＡＰＳにプッシュし、ＡＰＳは３－１のようになる。

ＣＲＦの６(0)、６(1)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込み、ＣＲＦは６－１のようになる。ここで、本動作例においては、終始分岐タグとして命

令デコード･設定ユニットから０が送られてくるものとする。

設定ポインタの値は０であるので、ＩＱのエントリ５(0)に上記命令の内容を書き込み、ＩＱは５－１のようになる。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈０〉,〈１〉を各々第１、第２デスティネーション･フィールドに書き込んでいる。さらに、設定ポインタの値に１を加え１にする。ここで、本動作例においては、ＩＱのＳフィールドには、命令が未実行であれば０、実行済み／正常終了あるいはストア命令におけるストア実行可能であれば１が書き込まれるものとする。

ＩＱのエントリ５(0)に書き込まれるものと同じ上記命令の内容を、ＩＱのエントリのアドレス０と共にＬＳＵに送る。

（１－Ｂ）第１サイクルの実行ステージ

当初のＩＱにおいては、実行可能な命令が書き込まれているエントリは存在しないので、実行ステージの動作としては何も行われない。

（１－Ｃ）第１サイクルの完了ステージ

当初のＩＱの完了ポインタが示すエントリ５(0)において、命令はまだ書き込まれていないため、完了ステージの動作としては何も行われない。

（２－Ａ）第２サイクルの命令デコード･設定ステージ

命令 **dload** ［B］のデコード･設定を行う。倍長語の変数データのオペランド･スタックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦの２エントリ６(2)、６(3)をそのデータを保持すべく割り付け、そのエントリのアドレス〈２〉,〈３〉をＡＰＳにプッシュし、ＡＰＳは３－２のようになる。

ＣＲＦの６(2)、６(3)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込み、ＣＲＦは６－

２のようになる。

設定ポインタの値は１であるので、ＩＱのエントリ５(1)に上記命令の内容を書き込み、ＩＱは５－２のようになる。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈２〉,〈３〉を各々第１、第２デスティネーション･フィールドに書き込んでいる。さらに、設定ポインタの値に１を加え２にする。

ＩＱのエントリ５(1)に書き込まれるものと同じ上記命令の内容を、ＩＱのエントリのアドレス１と共にＬＳＵに送る。

（２－Ｂ）第２サイクルの実行ステージ

ＬＳＵは変数Ａのアクセスを開始する。レイテンシは２サイクルであるとする。

（２－Ｃ）第２サイクルの完了ステージ

５－１の状態にあるＩＱの完了ポインタが示すエントリ５(0)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

（３－Ａ）第３サイクルの命令デコード･設定ステージ

命令 **dadd** のデコード･設定を行う。オペランド･スタックから４語のソース･データをポップして演算を行い、倍長語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈０〉,〈１〉,〈２〉,〈３〉をポップし、ＦＬに登録されているフリーなＣＲＦの２エントリ６(4)、６(5)を演算結果を保持すべく割り付け、そのエントリのアドレス〈４〉,〈５〉をＡＰＳにプッシュし、ＡＰＳは３－３のようになる。

ＣＲＦの６(4)、６(5)の各エントリにおいては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

設定ポインタの値は２であるので、ＩＱのエントリ５(2)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリ

のアドレス〈0〉,〈1〉,〈2〉,〈3〉を各々第1～第4ソース・フィールドに、APSにプッシュされる〈4〉,〈5〉を各々第1、第2デスティネーション・フィールドに書き込んでいる。また、6－2の状態にあるCRFの6(0)、6(1)、6(2)、6(3)の各エントリでWCFフィールドを読み出し、それぞれIQのWCF1～4の各フィールドに送っている。さらに、設定ポインタの値に1を加え3にする。

（3－B）第3サイクルの実行ステージ

LSUは変数Bのアクセスを開始する。レイテンシは2サイクルであるとする。

LSUから変数Aのデータを構成する2語 A_1、A_2 が送られてくるので、CRFのエントリ6(0)、6(1)のデータ・フィールドに各々書き込み、WCFフィールドを1に変更する。その上、IQにおいて各ソース・フィールドでCRFのエントリのアドレス〈0〉,〈1〉を照合するが、この場合、同じタイミングで命令が書き込まれるIQのエントリ5(2)の第1、第2ソース・フィールドで一致するので、同じエントリのWCF1、2の各フィールドを1とする。(WCF3、4の各フィールドについては、（3－A）で説明したように6－2の状態にあるCRFの対応するエントリのWCFフィールドが読み出され送られてくるので、これが書き込まれる。)

以上のようにIQのエントリ5(0)に書き込まれた命令の実行が正常に終了するので、5(0)のSフィールドを正常終了を意味する1に変更する。

（3－C）第3サイクルの完了ステージ

5－2の状態にあるIQの完了ポインタが示すエントリ5(0)において、Sフィールドは0であるので、完了ステージの動作としては何も行われない。

（４－Ａ）第４サイクルの命令デコード･設定ステージ

命令 d2f のデコード･設定を行う。オペランド･スタックから２語のソース･データをポップして変換演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈４〉,〈５〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(6)を演算結果を保持すべく割り付け、そのエントリのアドレス〈６〉をＡＰＳにプッシュし、ＡＰＳは３－４のようになる。

ＣＲＦのエントリ６(6)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

設定ポインタの値は３であるので、ＩＱのエントリ５(3)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈４〉,〈５〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈６〉を第１デスティネーション･フィールドに書き込んでいる。また、６－３の状態にあるＣＲＦの６(4)、６(5)の各エントリでＷＣＦフィールドを読み出し、それぞれＩＱのＷＣＦ１、２の各フィールドに送っている。さらに、設定ポインタの値に１を加え４にする。

（４－Ｂ）第４サイクルの実行ステージ

ＬＳＵから変数Ｂのデータを構成する２語 B_1、B_2 が送られてくるので、ＣＲＦのエントリ６(2)、６(3)のデータ･フィールドに各々書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈２〉,〈３〉を照合するが、この場合、ＩＱのエントリ５(2)の第３、第４ソース･フィールドで一致するので、同じエントリのＷＣＦ３、４の各フィールドを１とする。

以上のようにＩＱのエントリ５(1)に書き込まれた命令の実行が正常に終了するので、５(1)のＳフィールドを正常終了を意味する１に変更

する。

（４－Ｃ）第４サイクルの完了ステージ

５－３の状態にあるＩＱの完了ポインタが示すエントリ５(0)において、Ｓフィールドが１となったので、５(0)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、ＩＱのエントリ５(0)のデスティネーション･フィールドに書き込まれている〈０〉,〈１〉をＣＰＳにプッシュし、ＣＰＳは４－４のようになる。さらに、完了ポインタの値に１を加え１とし、これで、５(0)の命令は完了したことになる。

（５－Ａ）第５サイクルの命令デコード･設定ステージ

命令 fload ［T］のデコード･設定を行う。１語の変数データのオペランド･スタックへのロード命令であるので、ＦＬに登録されているフリーなＣＲＦのエントリ６(7)をそのデータを保持すべく割り付け、そのエントリのアドレス〈７〉をＡＰＳにプッシュし、ＡＰＳは３－５のようになる。

ＣＲＦのエントリ６(7)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

設定ポインタの値は４であるので、ＩＱのエントリ５(4)に上記命令の内容を書き込む。この際、ＡＰＳにプッシュされるＣＲＦのエントリのアドレス〈７〉を第１デスティネーション･フィールドに書き込んでいる。さらに、設定ポインタの値に１を加え５にする。

ＩＱのエントリ５(4)に書き込まれるものと同じ上記命令の内容を、ＩＱのエントリのアドレス４と共にＬＳＵに送る。

（５－Ｂ）第５サイクルの実行ステージ

５－４の状態にあるＩＱにおいて、演算命令が書き込まれているエントリ５(2)はＷＣＦフィールドが全て１となっているので、このエントリ５(2)の内容をそのエントリのアドレス２及びＣＲＦの６(0)、６(1)、

6(2)、6(3)の各エントリに書き込まれているソース・データと共に演算ユニット0に送り演算を開始させる。この演算のレイテンシは2サイクルであるとする。

（5－C）第5サイクルの完了ステージ

5－4の状態にあるIQの完了ポインタが示すエントリ5(1)において、Sフィールドが1となったので、5(1)の内容に基づいてCPS（及びCRF）を操作する。すなわち、IQのエントリ5(1)のデスティネーション・フィールドに書き込まれている〈2〉,〈3〉をCPSにプッシュし、CPSは4－5のようになる。さらに、完了ポインタの値に1を加え2とし、これで、5(1)の命令は完了したことになる。

（6－A）第6サイクルの命令デコード・設定ステージ

命令 dup_x1 のデコード・設定を行う。命令 dup_x1 は、ワード・スタックが、(右方向に成長するものとして) ...., word1, word2 のような状態であるとき、これを ...., word2, word1, word2 と変えるような、スタック上で1語のコピーを作成するオペランド・スタックの操作命令であるので、FLに登録されているフリーなCRFのエントリ6(8)をコピー・データを保持すべく割り付け、3－5のように下から〈6〉,〈7〉となっている状態のAPSを3－6のように〈8〉,〈6〉,〈7〉と変える。

CRFのエントリ6(8)においては、BBフィールドに1を立て、WCF及びCの各フィールドには0を書き込む。

設定ポインタの値は5であるので、IQのエントリ5(5)に上記命令の内容を書き込む。この際、コピー元となるデータを保持すべく割り付けられているCRFのエントリのアドレス〈7〉を第1ソース・フィールドに、コピー・データを保持すべく新たに割り付けられるCRFのエントリのアドレス〈8〉を第1デスティネーション・フィールドに書き込んでいる。また、6－5の状態にあるCRFのエントリ6(7)でWC

Ｆフィールドを読み出し、ＩＱのＷＣＦ１フィールドに送っている。さらに、設定ポインタの値に１を加え６にする。

（６－Ｂ）第６サイクルの実行ステージ

ＬＳＵは変数Ｔのアクセスを開始する。レイテンシは２サイクルであるとする。

演算ユニット０で５(2)の演算命令の実行が正常に終了すれば、演算結果を構成する２語 (A+B)_1、(A+B)_2 が送られてくるので、ＣＲＦのエントリ６(4)、６(5)のデータ･フィールドに各々書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈４〉,〈５〉を照合するが、この場合、ＩＱのエントリ５(3)の第１、第２ソース･フィールドで一致するので、同じエントリのＷＣＦ１、２の各フィールドを１とする。

以上のようにＩＱのエントリ５(2)に書き込まれた命令の実行が正常に終了するので、５(2)のＳフィールドを正常終了を意味する１に変更する。

（６－Ｃ）第６サイクルの完了ステージ

５－５の状態にあるＩＱの完了ポインタが示すエントリ５(2)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

（７－Ａ）第７サイクルの命令デコード･設定ステージ

命令 **fdiv** のデコード･設定を行う。オペランド･スタックから２語のソース･データをポップして演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈６〉,〈７〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(9)を演算結果を保持すべく割り付け、そのエントリのアドレス〈９〉をＡＰＳにプッシュし、ＡＰＳは３－７のようになる。

ＣＲＦのエントリ６(9)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

設定ポインタの値は６であるので、ＩＱのエントリ５(6)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈6〉,〈7〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈9〉を第１デスティネーション･フィールドに書き込んでいる。また、６－６の状態にあるＣＲＦの６(6)、６(7)の各エントリでＷＣＦフィールドを読み出し、それぞれＩＱのＷＣＦ１、２の各フィールドに送っている。さらに、設定ポインタの値に１を加え７にする。

（７－Ｂ）第７サイクルの実行ステージ

５－６の状態にあるＩＱにおいて、演算命令が書き込まれているエントリ５(3)は有効なＷＣＦフィールドが全て１となっているので、このエントリ５(3)の内容をそのエントリのアドレス３及びＣＲＦの６(4)、６(5)の各エントリに書き込まれているソース･データと共に演算ユニット０に送り演算を開始させる。この演算のレイテンシは２サイクルであるとする。

ＬＳＵから変数Ｔのデータが送られてくるので、ＣＲＦのエントリ６(7)のデータ･フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈7〉を照合するが、この場合、ＩＱのエントリ５(5)の第１ソース･フィールド及び同じタイミングで命令が書き込まれる５(6)の第２ソース･フィールドで一致するので、５(5)のＷＣＦ１フィールド及び５(6)のＷＣＦ２フィールドを１とする。（５(6)のＷＣＦ１フィールドについては、（７－Ａ）で説明したように６－６の状態にあるＣＲＦのエントリ６(6)のＷＣＦフィールドが読み出され送られてくるので、これが書

き込まれる。）

以上のようにＩＱのエントリ５(4)に書き込まれた命令の実行が正常に終了するので、５(4)のＳフィールドを正常終了を意味する１に変更する。

（７－Ｃ）第７サイクルの完了ステージ

５－６の状態にあるＩＱの完了ポインタが示すエントリ５(2)において、Ｓフィールドが１となったので、５(2)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＱのエントリ５(2)のソース･フィールドに書き込まれている〈０〉，〈１〉，〈２〉，〈３〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈４〉，〈５〉をＣＰＳにプッシュし、ＣＰＳは４－７のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(0)、６(1)、６(2)、６(3)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈０〉，〈１〉，〈２〉，〈３〉をＦＬに登録する。さらに、完了ポインタの値に１を加え３とし、これで、５(2)の命令は完了したことになる。

（８－Ａ）第８サイクルの命令デコード･設定ステージ

命令 **fsub** のデコード･設定を行う。オペランド･スタックから２語のソース･データをポップして演算を行い、１語の演算結果をプッシュする演算命令であるので、ＡＰＳから〈８〉，〈９〉をポップし、ＦＬに登録されているフリーなＣＲＦのエントリ６(10)を演算結果を保持すべく割り付け、そのエントリのアドレス〈１０〉をＡＰＳにプッシュし、ＡＰＳは３－８のようになる。

ＣＲＦのエントリ６(10)においては、ＢＢフィールドに１を立て、ＷＣＦ及びＣの各フィールドには０を書き込む。

設定ポインタの値は７であるので、ＩＱのエントリ５(7)に上記命令

の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈８〉,〈９〉を各々第１、第２ソース･フィールドに、ＡＰＳにプッシュされる〈１０〉を第１デスティネーション･フィールドに書き込んでいる。また、６－７の状態にあるＣＲＦの６(8)、６(9)の各エントリでＷＣＦフィールドを読み出し、それぞれＩＱのＷＣＦ１、２の各フィールドに送っている。さらに、設定ポインタの値に１を加え８にする。

（８－Ｂ）第８サイクルの実行ステージ

演算ユニット０で５(3)の変換演算命令の実行が正常に終了すれば、１語の演算結果（A+B）が送られてくるので、ＣＲＦのエントリ６(6)のデータ･フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈６〉を照合するが、この場合、ＩＱのエントリ５(6)の第１ソース･フィールドで一致するので、同じエントリのＷＣＦ１フィールドを１とする。

以上のようにＩＱのエントリ５(3)に書き込まれた命令の実行が正常に終了するので、５(3)のＳフィールドを正常終了を意味する１に変更する。

５－７の状態にあるＩＱにおいて、スタック上でコピーを作成するようなオペランド･スタックの操作命令が書き込まれているエントリ５(5)は有効なＷＣＦフィールドが１となっているので、データのコピーを実行する。すなわち、ＣＲＦのエントリ６(7)からデータを読み出し、これをＣＲＦのエントリ６(8)のデータ･フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈８〉を照合するが、この場合、同じタイミングで命令が書き込まれるＩＱのエントリ５(7)の第１ソース･フィ

ールドで一致するので、同じエントリのＷＣＦ１フィールドを１とする。（ＷＣＦ２フィールドについては、（８－Ａ）で説明したように６－７の状態にあるＣＲＦのエントリ６(9)のＷＣＦフィールドが読み出され送られてくるので、これが書き込まれる。）ＩＱのエントリ５(5)のＳフィールドを正常終了を意味する１に変更する。

（８－Ｃ）第８サイクルの完了ステージ

５－７の状態にあるＩＱの完了ポインタが示すエントリ５(3)において、Ｓフィールドは０であるので、完了ステージの動作としては何も行われない。

（９－Ａ）第９サイクルの命令デコード･設定ステージ

命令 fstore ［X］のデコード･設定を行う。スタックトップにある１語のデータのストア命令であるので、ＡＰＳから〈１０〉をポップし、ＡＰＳは３－９のようになる。

設定ポインタの値は８であるので、ＩＱのエントリ５(8)に上記命令の内容を書き込む。この際、ＡＰＳからポップされるＣＲＦのエントリのアドレス〈１０〉を第１ソース･フィールドに書き込んでいる。また、６－８の状態にあるＣＲＦのエントリ６(10)でＷＣＦフィールドを読み出し、ＩＱのＷＣＦ１フィールドに送っている。さらに、設定ポインタの値に１を加え９にする。

ＩＱのエントリ５(8)に書き込まれるものと同じ上記命令の内容を、ＩＱのエントリのアドレス８と共にＬＳＵに送る。

（９－Ｂ）第９サイクルの実行ステージ

５－８の状態にあるＩＱにおいて、演算命令が書き込まれているエントリ５(6)は有効なＷＣＦフィールドが全て１となっているので、このエントリ５(6)の内容をそのエントリのアドレス６及びＣＲＦの６(6)、６(7)の各エントリに書き込まれているソース･データと共に演算ユニッ

ト１に送り演算を開始させる。この演算のレイテンシは１０サイクルであるとする。

（９－Ｃ）第９サイクルの完了ステージ

５－８の状態にあるＩＱの完了ポインタが示すエントリ５(3)において、Ｓフィールドが１となったので、５(3)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＱのエントリ５(3)のソース・フィールドに書き込まれている〈４〉，〈５〉をＣＰＳからポップし、デスティネーション・フィールドに書き込まれている〈６〉をＣＰＳにプッシュし、ＣＰＳは４－９のようになる。エントリ・アドレスがＣＰＳからポップされるＣＲＦの６(4)、６(5)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈４〉，〈５〉をＦＬに登録する。さらに、完了ポインタの値に１を加え４とし、これで、５(3)の命令は完了したことになる。

（１０－Ｃ）第１０サイクルの完了ステージ

５－９の状態にあるＩＱの完了ポインタが示すエントリ５(4)において、Ｓフィールドが１であるので、５(4)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、ＩＱのエントリ５(4)のデスティネーション・フィールドに書き込まれている〈７〉をＣＰＳにプッシュし、ＣＰＳは４－１０のようになる。さらに、完了ポインタの値に１を加え５とし、これで、５(4)の命令は完了したことになる。

（１１－Ｃ）第１１サイクルの完了ステージ

５－１０の状態にあるＩＱの完了ポインタが示すエントリ５(5)において、Ｓフィールドが１であるので、５(5)の内容に基づいてＣＰＳ（及びＣＲＦ）を操作する。すなわち、（６－Ａ）におけるＡＰＳの動作が再現され、ＣＰＳは４－１１のようになる。さらに、完了ポインタの値に１を加え６とし、これで、５(5)の命令は完了したことになる。

（１８－Ｂ）第１８サイクルの実行ステージ

演算ユニット１で５(6)の演算命令の実行が正常に終了すれば、１語の演算結果（A+B)/T　が送られてくるので、ＣＲＦのエントリ６(9)のデータ･フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース･フィールドでＣＲＦのエントリのアドレス〈9〉を照合するが、この場合、ＩＱのエントリ５(7)の第２ソース･フィールドで一致するので、同じエントリのＷＣＦ２フィールドを１とする。

以上のようにＩＱのエントリ５(6)に書き込まれた命令の実行が正常に終了するので、５(6)のＳフィールドを正常終了を意味する１に変更する。

（１９－Ｂ）第１９サイクルの実行ステージ

５－１８の状態にあるＩＱにおいて、演算命令が書き込まれているエントリ５(7)は有効なＷＣＦフィールドが全て１となっているので、このエントリ５(7)の内容をそのエントリのアドレス７及びＣＲＦの６(8)、６(9)の各エントリに書き込まれているソース･データと共に演算ユニット０に送り演算を開始させる。この演算のレイテンシは２サイクルであるとする。

（１９－Ｃ）第１９サイクルの完了ステージ

５－１８の状態にあるＩＱの完了ポインタが示すエントリ５(6)において、Ｓフィールドが１となったので、５(6)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＱのエントリ５(6)のソース･フィールドに書き込まれている〈6〉,〈7〉をＣＰＳからポップし、デスティネーション･フィールドに書き込まれている〈9〉をＣＰＳにプッシュし、ＣＰＳは４－１９のようになる。エントリ･アドレスがＣＰＳからポップされるＣＲＦの６(6)、６(7)の各エントリでは、ＢＢフィー

ルドを０に変更する。ＣＲＦのエントリのアドレス〈６〉，〈７〉をＦＬに登録する。さらに、完了ポインタの値に１を加え７とし、これで、５(6)の命令は完了したことになる。

（２０－Ｂ）第２０サイクルの実行ステージ

演算ユニット０で５(7)の演算命令の実行が正常に終了すれば、１語の演算結果 T-(A+B)/T が送られてくるので、ＣＲＦのエントリ６(10)のデータ・フィールドに書き込み、ＷＣＦフィールドを１に変更する。その上、ＩＱにおいて各ソース・フィールドでＣＲＦのエントリのアドレス〈１０〉を照合するが、この場合、ＩＱのエントリ５(8)の第１ソース・フィールドで一致するので、同じエントリのＷＣＦ１フィールドを１とする。

以上のようにＩＱのエントリ５(7)に書き込まれた命令の実行が正常に終了するので、５(7)のＳフィールドを正常終了を意味する１に変更する。

（２１－Ｂ）第２１サイクルの実行ステージ

５－２０の状態にあるＩＱにおいて、ストア命令が書き込まれているエントリ５(8)は有効なＷＣＦフィールドが１となっているので、ＩＱのエントリのアドレス８と共にＣＲＦのエントリ６(10)に書き込まれているソース・データをＬＳＵに送る。５(8)のＳフィールドをストア実行可能を意味する１に変更する。

（２１－Ｃ）第２１サイクルの完了ステージ

５－２０の状態にあるＩＱの完了ポインタが示すエントリ５(7)において、Ｓフィールドが１となったので、５(7)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＱのエントリ５(7)のソース・フィールドに書き込まれている〈８〉，〈９〉をＣＰＳからポップし、デスティネーション・フィールドに書き込まれている〈１０〉をＣＰＳにプ

ッシュし、ＣＰＳは４－２１のようになる。エントリ・アドレスがＣＰＳからポップされるＣＲＦの６(8)、６(9)の各エントリでは、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈８〉,〈９〉をＦＬに登録する。さらに、完了ポインタの値に１を加え８とし、これで、５(7)の命令は完了したことになる。

（２２－Ｃ）第２１サイクルの完了ステージ

５－２１の状態にあるＩＱの完了ポインタが示すエントリ５(8)においては、ストア命令が書き込まれており、Ｓフィールドが１となったので、ＬＳＵにＩＱのエントリのアドレス８を示して、データ・キャッシュへのストアの実行を依頼する。さらに、５(8)の内容に基づいてＣＰＳ及びＣＲＦを操作する。すなわち、ＩＱのエントリ５(8)のソース・フィールドに書き込まれている〈１０〉をＣＰＳからポップし、ＣＰＳは４－２２のようになる。エントリ・アドレスがＣＰＳからポップされるＣＲＦのエントリ６(10)では、ＢＢフィールドを０に変更する。ＣＲＦのエントリのアドレス〈１０〉をＦＬに登録する。さらに、完了ポインタの値に１を加え９とし、これで、５(8)の命令は完了したことになる。

以上で、本第１実施例の計算機システムにおいて X=T-(A+B)/T の計算が完了したことになる。

本発明の計算機システムにおいては、分岐予測に基づく投機的実行を実現することができる。ＡＰＳ履歴ファイルは、投機的実行を可能にするために具備されるものである。条件分岐命令がデコードされるごとに、ＡＰＳ履歴ファイルの１エントリにＡＰＳの全エントリ及び PP_OF_APS の内容を書き込むようになっている。以下に、本実施例の計算機システムにおいて、分岐予測に基づく投機的実行がどのように行われるかについて説明する。

前述のように、本実施例の計算機システムにおいては、命令デコード

・設定ステージにおいて、命令をデコードして、命令の内容に応じてＡＰＳ及びＣＲＦを操作すると共に、命令の内容をＩＱに書き込むようになっている。初期状態から命令が流れ始め最初の条件分岐命令がデコードされるまでの間、デコードされる命令に分岐タグとして０を付し、この分岐タグ０を、命令の内容が書き込まれるＩＱのエントリのＢＴフィールド、及び、割り付けられるＣＲＦのエントリのＣフィールドに書き込む。

最初の条件分岐命令がデコードされ分岐予測が行われる際に、分岐時点の状態を保存するために、ＡＰＳの全エントリ及び PP_OF_APS の内容をＡＰＳ履歴ファイルのアドレス０のエントリに書き込む。上記の分岐予測に基づいた命令の流れにおいては、分岐タグとして１を付し、ＩＱ及びＣＲＦの設定を行う。

２つ目の条件分岐命令がデコードされた時に、最初の条件分岐命令が未確定である場合、あるいは確定して予測が当たっていた場合には、ＡＰＳの全エントリ及び PP_OF_APS の内容をＡＰＳ履歴ファイルのアドレス１のエントリに書き込む。２段目の分岐予測に基づいた命令の流れにおいては、分岐タグとして２を付し、ＩＱ及びＣＲＦの設定を行う。

分岐予測が当たり続ければ以後同様に処理が進み、ＡＰＳ履歴ファイルへの書き込みはアドレス順に行われる。また、ＡＰＳ履歴ファイルのアドレス n のエントリに書き込みが行われてから次に書き込みが行われるまでの間にデコードされる命令には分岐タグとして n+1 を付すものとする。

分岐予測が外れた場合には、その条件分岐命令以降にデコードされた命令に付された分岐タグをもとに、演算ユニット、分岐ユニット及びＬＳＵの各実行ユニットに送られた命令をキャンセルし、ＣＲＦにおいてＣフィールドで分岐タグを照合してその一致するエントリの各々に対し

て、そのＢＢフィールドを０に変更して、そのエントリのアドレスをＦＬに登録し、ＩＱの設定ポインタの値をその条件分岐命令が書き込まれているエントリの次のアドレスに書き換えることによって、その条件分岐命令以降にデコード･設定された命令を無効とする。さらに、同じエントリ･アドレスにあるＣＰＳのエントリとその内容が一致しないＡＰＳの各エントリ及び PP_OF_APS に、その条件分岐命令がデコードされた際にＡＰＳ履歴ファイルに書き込まれた内容をコピーして、正しい位置の命令から処理を再開する。

以上のように、本発明の計算機システムにおいては、ＡＰＳ履歴ファイルを用いることによって、条件分岐命令がデコードされ分岐予測が行われる各々の時点の状態を再構成することができるので、分岐予測に基づく投機的実行が可能である。

以上では、説明を簡単にするため、１サイクルで同時にデコード･設定／完了できる命令は高々１つまでとして説明してきた。本発明の計算機システムにおいては、同時に複数の命令をデコード･設定／完了できる構成とすることができる。すなわち、ＦＬがＦＩＦＯキューの構成となっていれば、割り付けのためにフリーなＣＲＦのエントリのアドレスをＦＬから取り出す順番は決まっており、各命令における何語ポップし何語プッシュするかというようなスタック操作の内容を把握して、同時に複数の命令をデコード･設定することができる。また、命令の完了の動作においては、各命令におけるＣＰＳに対するスタック操作があらかじめ厳密に決められているので、より容易に複数命令の同時完了を実現することができる。

同時にデコード･設定／完了できる命令の数を多くするほど、命令デコード･設定ユニットその他の制御回路が複雑になると共に、ＩＱやＣＲＦを構成する各レジスタ･ファイルのポートの数やＩＱの各エントリ

のソース・フィールドごとに設けられる比較回路の数、演算ユニットの数、さらに構成要素間を結合するバスの数などの点で、より多量のハードウェアが必要となる。

本発明の計算機システムにおいては、デコード・設定を２つのステージに分けて行うこととし、その前半のステージにおいて、同時にデコード・設定する複数の命令の内容を統合した形式に変換するような構成とすることも可能である。

たとえば、１サイクル当り３命令までデコード・設定できるような構成をとる場合、前述の X=T-(A+B)/T を計算するプログラムは第１３図の図表に示されるような内容に変換される。第１３図の図表の各段には、同時にデコード・設定される３つの命令に基づく、PP_OF_APS の増分、ＡＰＳの操作内容及びＩＱの３エントリに書き込まれるべき設定内容を示している。ここでは、設定前のＡＰＳの内容を ..... s2, s1, s0 （右端がスタックトップ）、ＦＩＦＯキューの構成となっているフリー・リストの内容を（取り出される順に）f1, f2, f3 ..... として記述しており、デコード・設定の後半のステージにおいて、それぞれ対応するＣＲＦのエントリ・アドレスがＡＰＳ／ＩＱに書き込まれるようになっている。PP_OF_APS の増分の欄で示されるようにＡＰＳのスタックトップの位置が移動するが、ＡＰＳの操作内容の欄では、この移動後のスタックトップの位置が右端に対応している。また、'NC' は「変化なし(No Change)」を意味する。

ついで、本発明第２実施例の計算機システムについて説明する。

第２実施例は、オペランド・スタックの操作命令の処理方法が、第１実施例と異なる。

第２実施例の計算機システムは、第１実施例とは、統合レジスタ・フ

ァイル（ＣＲＦ）６のエントリの構成が異なるが、計算機システムの基本構成、及び、前進ポインタ・スタック（ＡＰＳ）３、完了ポインタ・スタック（ＣＰＳ）４、命令キュー（ＩＱ）５の構成は同様である。

第１４図は、本第２実施例における、ＣＲＦ６の各々のエントリ６(i)の詳細な構成を示す説明図である。ここで、ｉはエントリのアドレスである。ＣＲＦ６の各々のエントリ６(i)はデータ・フィールド６１(i)、書込み完了フラグ（ＷＣＦ，Write Completion Flag）フィールド６２(i)、カラー（Ｃ，Colour）フィールド６３(i)、ビジービット（ＢＢ）フィールド６４(i)、及び参照数（ＮＲ，Number of Reference）フィールド６５(i)から成っている。

ＣＲＦの各々のエントリの、データ・フィールド及びＷＣＦ、Ｃ、ＢＢの各フィールドは第１実施例と同様である。

ＣＲＦの各々のエントリにおいて、ＮＲフィールドは、そのＣＲＦのエントリのアドレスを保持しているＣＰＳのエントリの数が書き込まれているようになっている。

すなわち、ＣＲＦとデータ・キャッシュの間の Spill/Fill の動作及び命令の完了に基づくＣＰＳに対する操作に伴い、ＣＲＦの関係するエントリにおいて、そのエントリのアドレスを保持するＣＰＳのエントリの数の更新を行うようになっている。

第２実施例においては、ＣＲＦからデータ・キャッシュへの１語のデータのストア（Spill）は以下のように行われる。この場合、BP_OF_PS で示されるエントリの内容がＡＰＳとＣＰＳで一致していなければならないが、その２つのポインタ・スタックで一致する BP_OF_PS で示されるエントリの内容で示されるＣＲＦのエントリに書き込まれている１語分のデータをデータ・キャッシュにストア（Spill）し、BP_OF_PS の値に１を加える。さらに、上記ＣＲＦのエントリにおいて，ＮＲフィール

ドの値から１を引く。その結果、その値が０になれば、そのＣＲＦのエントリのＢＢフィールドを０に変更し、そのエントリのアドレスをＦＬに登録する。

逆に、データ・キャッシュからＣＲＦへの１語のデータのロード（Fill）は以下のように行われる。すなわち、最後にストア（Spill）した１語分のデータをデータ・キャッシュから取り出し、それにＦＬに登録されているフリーなＣＲＦの１エントリを割り付け、そのデータ・フィールドに書き込む。ＷＣＦ、ＮＲ、ＢＢの各フィールドは１とする。さらに、その割り付けられたＣＲＦのエントリのアドレスを、ＡＰＳ及びＣＰＳの BP_OF_PS に示されるエントリの１つ下に各々書き込み、BP_OF_PS の値から１を引く。

命令の処理においても、本第２実施例の計算機システムは第１実施例と概ね同様であるが、以下に、各ステージごとに第１実施例との相違を明確にすることにより、本第２実施例の計算機システムの動作を説明する。

① 命令フェッチ・ステージ

第１実施例と同様。

② 命令デコード・設定ステージ

以下の点を除き、第１実施例と同様。

・命令のデコード・設定に伴い新たに割り付けられるＣＲＦのエントリにおいて、ＮＲフィールドに０を書き込む。

・オペランド・スタックの操作命令の場合、オペランド・スタックに対して行うべき操作をＡＰＳに対して全く同様に行う。その命令の内容をＩＱに書き込む際に、Ｓフィールドは実行済みとする。また、スタック上でコピーを作成するようなオペランド・スタックの操作命令（ Java VM における dup, dup2, dup_x1, dup2_x1, dup_x2 及び dup2_x2 ）の場合にも、

ＦＬに登録されているフリーなＣＲＦのエントリを新たに割り付けることはしないので、ソース・フィールド及びデスティネーション・フィールドへの書き込みは必要ない。

③ 実行ステージ

以下の点を除き、第１実施例と同様。

・オペランド・スタックの操作命令に関しては、スタック上でコピーを作成するような命令の場合も、実行ステージにおける動作としては、何も行わない。

④ 完了ステージ

以下の点を除き、第１実施例と同様。

・オペランド・スタックの操作命令の完了においては、スタック上でコピーを作成するような命令の場合も、オペランド・スタックに対して行うべき操作をＣＰＳに対して全く同様に行う（命令がデコード・設定された際のＡＰＳの動作を再現すべくＣＰＳが操作される、という観点からは第１実施例と同様）。

・命令の完了に基づくＣＰＳに対する操作に伴い、関係するＣＲＦのエントリのＮＲフィールドの値を増減させる。すなわち、エントリ・アドレスがＣＰＳにプッシュされるＣＲＦのエントリではＮＲフィールドの値を０から１に変更し、スタック上でコピーを作成するようなオペランド・スタックの操作命令の完了に伴いエントリ・アドレスがＣＰＳ上でコピーされるＣＲＦのエントリではＮＲフィールドの値に１を加え、エントリ・アドレスがＣＰＳからポップされるＣＲＦのエントリではＮＲフィールドの値から１を引く。

・命令の完了に基づくＣＰＳに対する操作に伴い、エントリ・アドレスがＣＰＳからポップされるＣＲＦのエントリでは、ＮＲフィールドの値が０になった場合にのみ、ＢＢフィールドを０に変更し、そのエントリ

・アドレスをＦＬに登録する。

以上が、本第２実施例の計算機システムの動作についての全般的な説明である。

第１５図～第２１図は、本第２実施例の計算機システムにおいて、前述の X=T-(A+B)/T の計算を行うプログラムを処理する際の動作をサイクル毎に示した説明図であり、図中ＣＲＦ６及びＩＱ５の各エントリの構成は、それぞれ第１４図、第５図のものと同じである。第１５図～第２１図で空白となっている箇所は、そのフィールドの内容に留意する必要が無いことを意味する。時系列で各構成要素の内容を示すために、各部の符号の後尾にハイフンと各サイクルに対応する数字、及び、第１実施例における動作例を示す第６図～第１２図における符号と区別するために文字 'a' を添えている。

本発明の計算機システムは、上述の実施例に限られるものではなく、細部の構成の異なる様々な実施例が存在する。たとえば、次のようなものをあげることができる。

①ＩＱが各々のエントリにソース・データも書き込まれるような構成となっており、ＩＱにおいて、対応するＷＣＦフィールドに１が立つのと同じタイミングでソース・データを書き込むようにしたもの。

②各実行ユニットの入力段にリザベーション・ステーションを設けた構成とし、命令デコード・設定ステージにおいて、個々の命令の内容をＩＱに書き込むと共にそれぞれ適切なリザベーション・ステーションへ送るようにしたもの（ＬＳＵに関しては、リザベーション・ステーションと共にストア・バッファを備えた構成としてもよい）。

③整数データ用／浮動小数点データ用、あるいは 32 ビット・データ用／ 64 ビット・データ用というようにデータ型別にＣＲＦ及びフリー・リ

ストを備え、さらに、それぞれに対応してひと揃いの実行ユニットを設けたもの。

これらの実施例の多くは、レジスタ・ベースのスーパースカラ・アーキテクチャにおいて同様な主旨のものを見出すことのできるものである。

命令セットがスタック型の命令及びレジスタ型の命令を共に含むような、本発明に基づく計算機システムも実現可能である。すなわち、前進ポインタ・スタック及び完了ポインタ・スタックに加えて、レジスタ番号にそれぞれ対応して設けられた各エントリに統合レジスタ・ファイルのエントリ・アドレスが書き込まれるようになっている前進レジスタ・マッピング・テーブル及び完了レジスタ・マッピング・テーブルを具備する構成とし、スタック型の命令に関しては前進／完了ポインタ・スタックを操作し、レジスタ型の命令に関しては前進／完了レジスタ・マッピング・テーブルをアクセスするようにする。この場合、前進ポインタ・スタック履歴ファイルの代わりに、各々のエントリに前進ポインタ・スタック及び前進レジスタ・マッピング・テーブル双方の内容が書き込まれるようになっている前進履歴ファイルを具備する必要がある。

産業上の利用可能性

以上のように、本発明の計算機システムは、正確な例外処理を保証しつつ、スタックマシンの機械語で記述されたプログラムを out-of-order で処理するものであるが、複数の演算ユニットによる並列処理やそれらのパイプライン化によって効率的な処理を行うことが可能であるという利点がある。

また、分岐予測に基づく投機的実行や、１サイクル当り複数命令のデコード・設定／完了の可能な構成とすることにより、さらなる高速化が可能である。

請 求 の 範 囲

1．データ・キャッシュ（１１）と

各々のエントリにデータが書き込まれるようになっている統合レジスタ・ファイル（６）と

各々のエントリに上記統合レジスタ・ファイル（６）のエントリのアドレスが書き込まれるようになっているスタックの構成となっている前進ポインタ・スタック（３）と

各々のエントリに個々の命令の内容が書き込まれるようになっているＦＩＦＯ（First In First Out）キューの構成となっている命令キュー（５）と

演算を実行するようになっている演算ユニット（８０，８１）と

上記データ・キャッシュ（１１）及び上記統合レジスタ・ファイル（６）にアクセスできるようになっているロード／ストア・ユニット（８３）とを具備し、

オペランド・スタックに対するポップ操作を含む命令がデコードされた場合には、ポップすべき語数と同じ数だけ上記統合レジスタ・ファイル（６）のエントリのアドレスを上記前進ポインタ・スタック（３）からポップし、

オペランド・スタックに対するプッシュ操作を含む命令がデコードされた場合には、プッシュすべき語数と同じ数だけ割り付けられていない上記統合レジスタ・ファイル（６）のエントリを割り付け、上記割り付けた上記統合レジスタ・ファイル（６）のエントリのアドレスを上記前進ポインタ・スタック（３）にプッシュし、

デコードされた命令の内容を、ポップ／プッシュ操作を伴う命令の場合にはポップ／プッシュされる上記統合レジスタ・ファイル（６）のエ

ントリのアドレスと共に、上記命令キュー（５）に書き込み、

上記命令キュー（５）に保持されている未実行の命令をデータ駆動の原理に基づき処理するようになっている、スタックマシンの機械語で記述されたプログラムを実行する計算機システム。

２．各々のエントリに上記統合レジスタ・ファイル（６）のエントリのアドレスが書き込まれるようになっているスタックの構成となっている完了ポインタ・スタック（４）を具備し、

上記命令キュー（５）の先頭のエントリにおいて保持されている命令の完了が可能である、あるいはそうなると、上記命令キュー（５）の上記先頭のエントリの内容に基づき、上記保持されている命令がデコードされた際の上記前進ポインタ・スタック（３）の動作を再現すべく上記完了ポインタ・スタック（４）を操作し、上記命令キュー（５）から上記先頭のエントリを除外し、

ポップ操作によって上記完了ポインタ・スタック（４）におけるアドレスの保持が無くなった上記統合レジスタ・ファイル（６）のエントリの割り付けを解除するようになっている請求項１記載の計算機システム。

３．割り付けられていない上記統合レジスタ・ファイル（６）のエントリのアドレスを保持するフリー・リストを具備し、

初期状態においては、上記統合レジスタ・ファイル（６）の全てのエントリのアドレスが上記フリー・リストに登録されており、

上記統合レジスタ・ファイル（６）のエントリを割り付ける必要がある場合に、上記フリー・リストから割り付けられていない上記統合レジスタ・ファイル（６）のエントリのアドレスを取り出し、

割り付けが解除された上記統合レジスタ・ファイル（６）のエントリのアドレスを上記フリー・リストに登録するようになっている請求項２

記載の計算機システム。

4．上記前進ポインタ・スタック（3）と上記完了ポインタ・スタック（4）が循環型のバッファの構成となっており、

上記前進ポインタ・スタック（3）及び上記完了ポインタ・スタック（4）で、上記統合レジスタ・ファイル（6）のエントリのアドレスを保持する最下位のエントリの内容が一致する場合には、上記前進ポインタ・スタック（3）及び上記完了ポインタ・スタック（4）において上記最下位のエントリにおける上記統合レジスタ・ファイル（6）のエントリのアドレスの保持を解除し、上記一致する内容で示される上記統合レジスタ・ファイル（6）のエントリに書き込まれているデータを上記データ・キャッシュ（11）にストア（Spill）することができるようになっており、

上記データ・キャッシュ（11）に最後にストア（Spill）したデータに対し、割り付けられていない上記統合レジスタ・ファイル（6）の1エントリを割り付け、上記データを書き込み、上記前進ポインタ・スタック（3）及び上記完了ポインタ・スタック（4）において、上記統合レジスタ・ファイル（6）のエントリのアドレスを保持する最下位のエントリの1つ下のエントリに上記データが書き込まれる上記統合レジスタ・ファイル（6）のエントリのアドレスを保持させることによって、上記最後にストア（Spill）したデータの上記統合レジスタ・ファイル（6）へのロード（Fill）を行えるようになっている請求項2記載の計算機システム。

5．各々のエントリに上記前進ポインタ・スタック（3）の内容が書き込まれるようになっている前進ポインタ・スタック履歴ファイル（3a）を具備し、

上記統合レジスタ・ファイル（6）が各々のエントリに分岐タグも書

き込まれるような構成となっており、

命令のデコードの際に、割り付けられる上記統合レジスタ・ファイル（６）のエントリに分岐タグを書き込むようになっており、

条件分岐命令がデコードされるごとに、上記前進ポインタ・スタック履歴ファイル（３ａ）の１エントリに上記前進ポインタ・スタック（３）の内容を書き込み、分岐タグを変更して、分岐予測に基づく投機的実行を行い、

分岐予測が外れた場合には、その条件分岐命令以降にデコードされた命令を無効にし、上記条件分岐命令以降にデコードされた命令に付した分岐タグが書き込まれている上記統合レジスタ・ファイル（６）のエントリの割り付けを解除し、上記条件分岐命令がデコードされた際に書き込みの行われた上記前進ポインタ・スタック履歴ファイル（３ａ）のエントリの内容を、上記前進ポインタ・スタック（３）にコピーして、正しい位置の命令から処理を再開することによって、

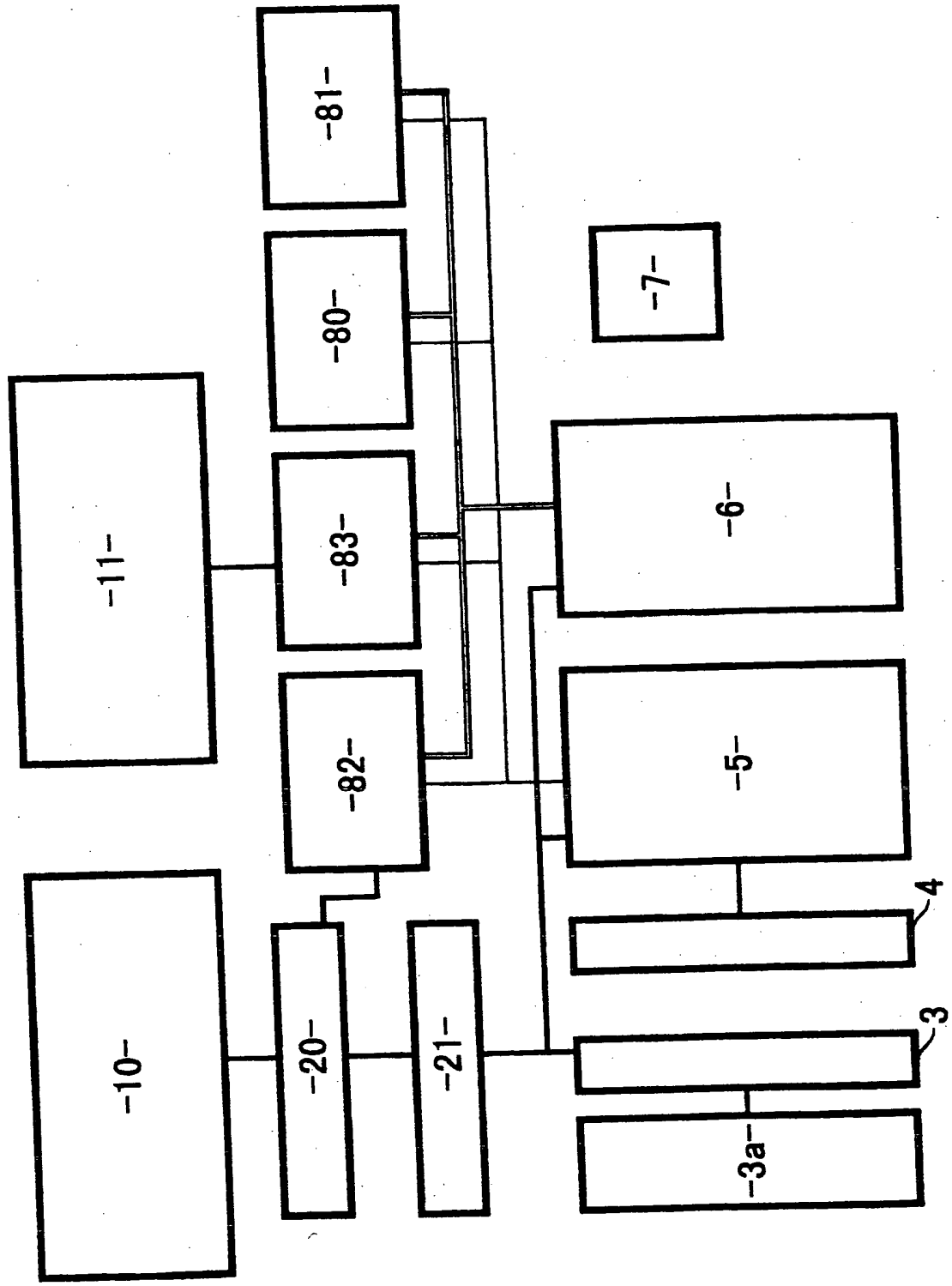
分岐予測に基づく投機的実行を行うようになっている請求項２記載の計算機システム。

６．上記フリー・リストがＦＩＦＯキューの構成となっており、

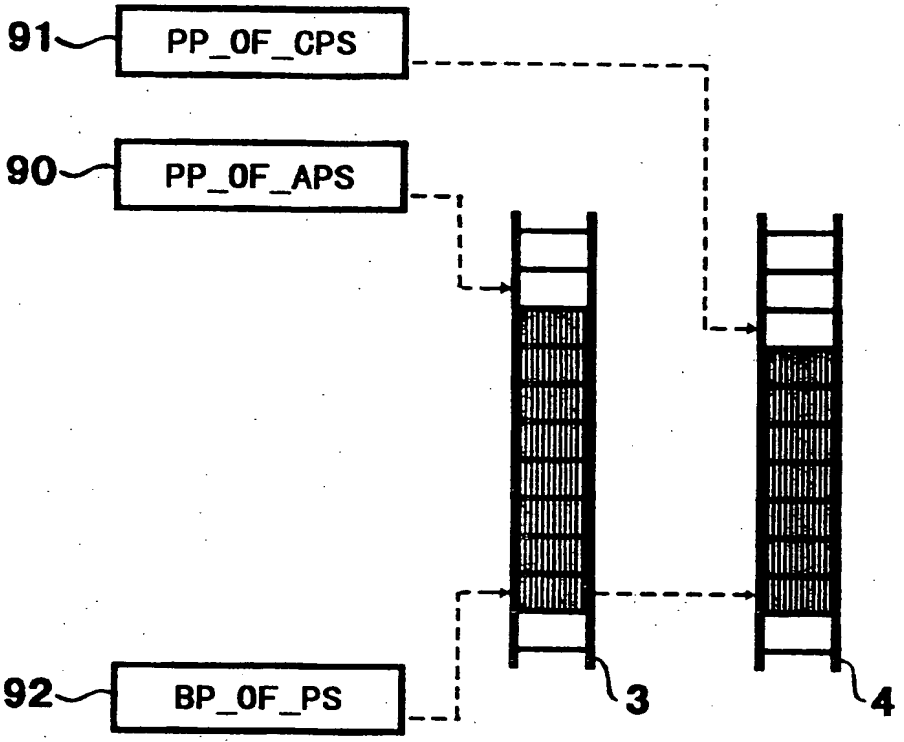
同時に複数の命令をデコードし、上記前進ポインタ・スタック（３）の操作、上記統合レジスタ・ファイル（６）のエントリの割り付け及び上記命令キュー（５）の連続する複数のエントリへの命令の内容の書き込みを行う機能と、

同時に上記命令キュー（５）の連続する複数のエントリに書き込まれている内容に基づき、上記完了ポインタ・スタック（４）の操作及び上記統合レジスタ・ファイル（６）のエントリの割り付けの解除を行う機能を有する請求項３記載の計算機システム。

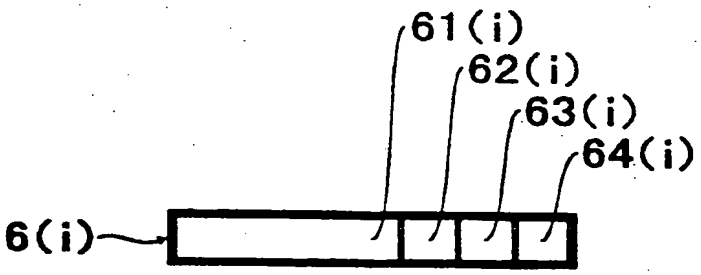
第 1 図



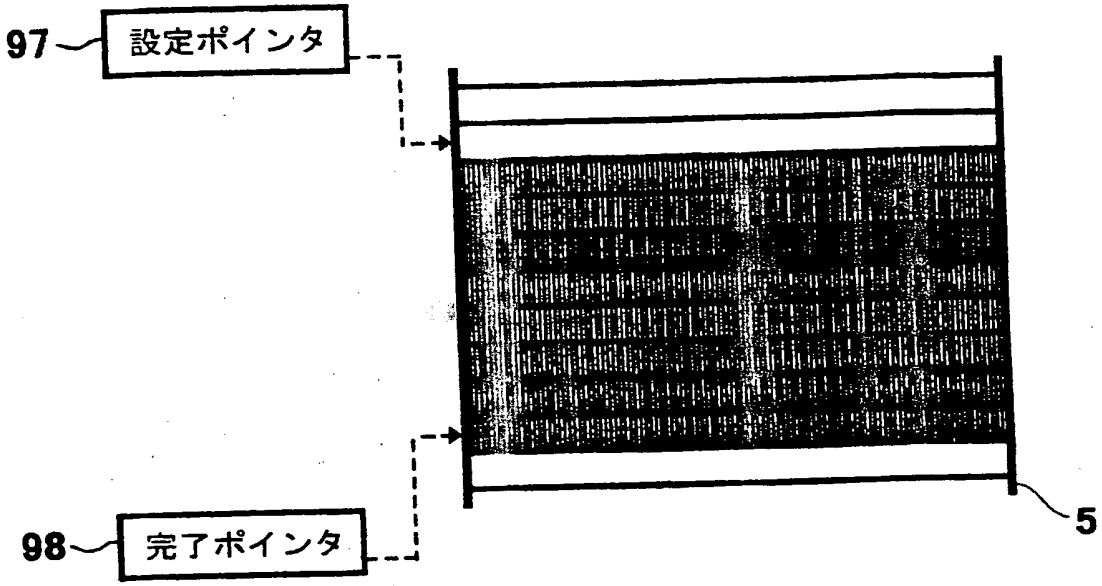
第 2 図



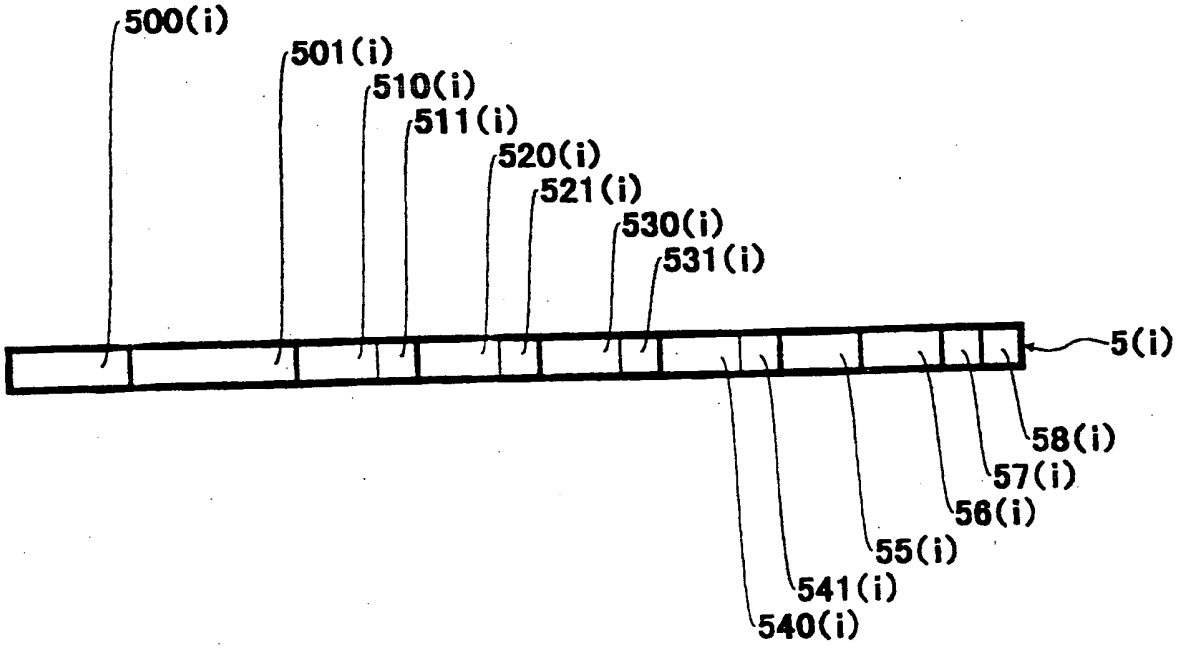
第 3 図



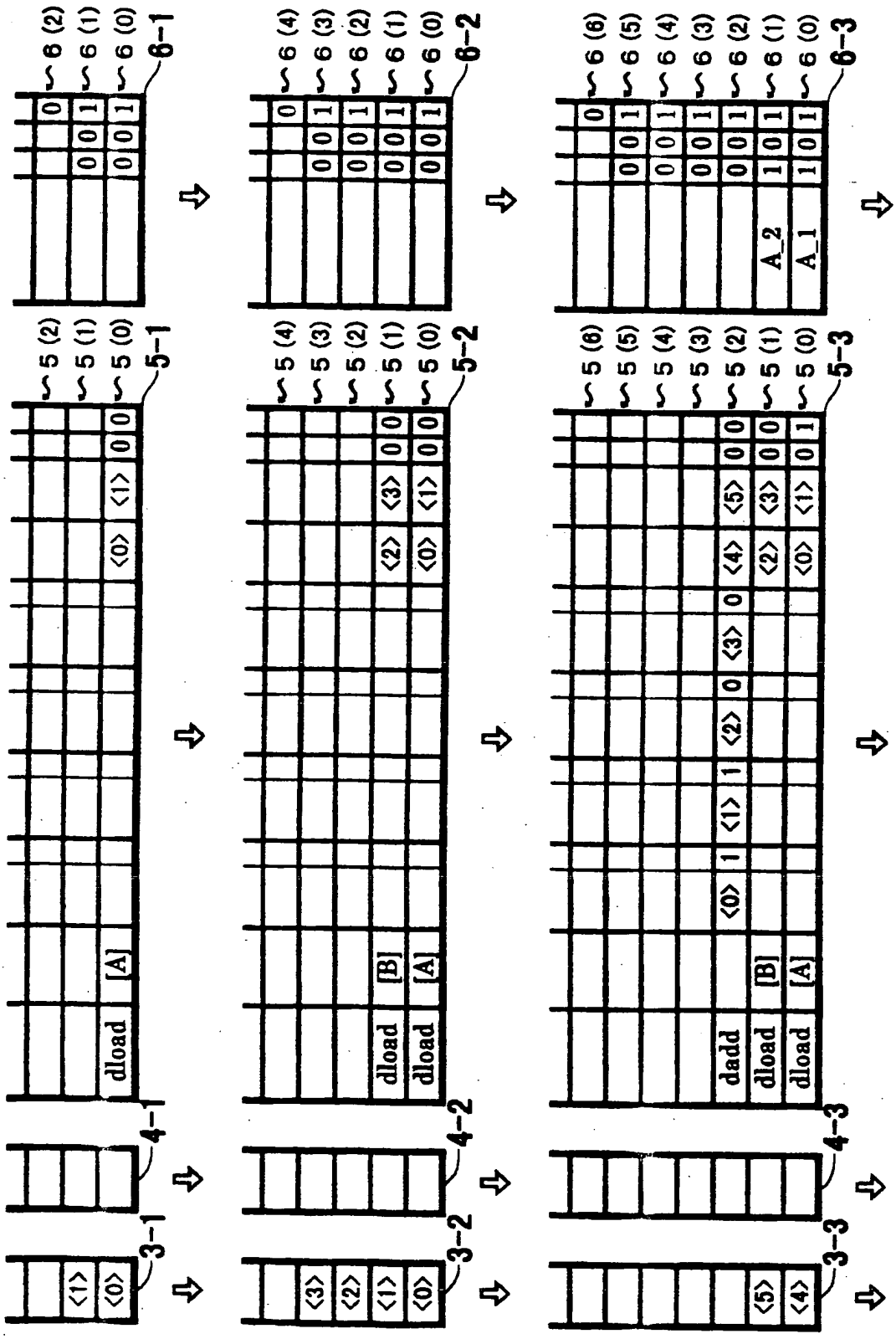
第 4 図


97
設定ポインタ
98
完了ポインタ
5

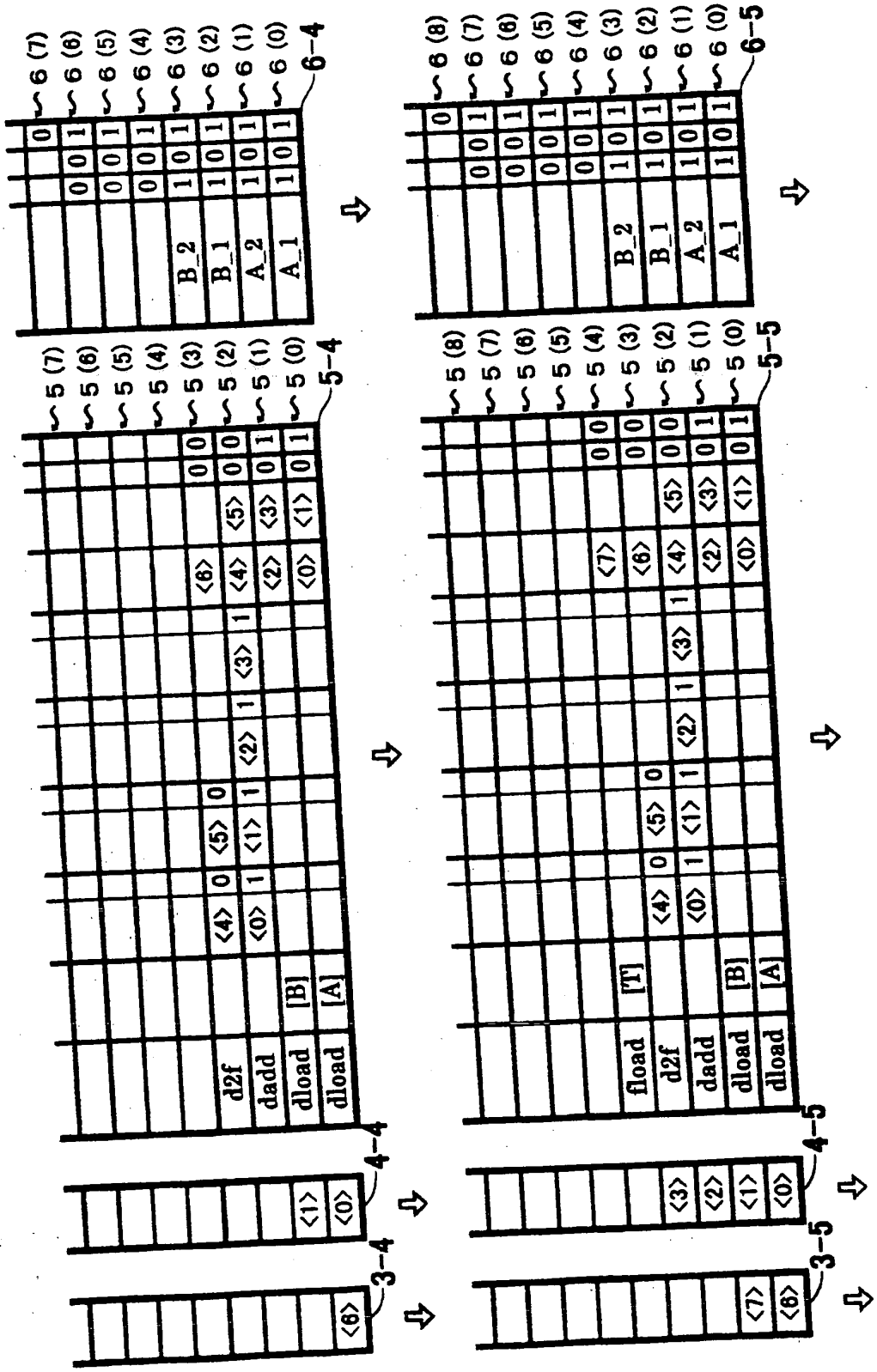
第 5 図


500(i)
501(i)
510(i)
511(i)
520(i)
521(i)
530(i)
531(i)
5(i)
58(i)
57(i)
56(i)
55(i)
541(i)
540(i)

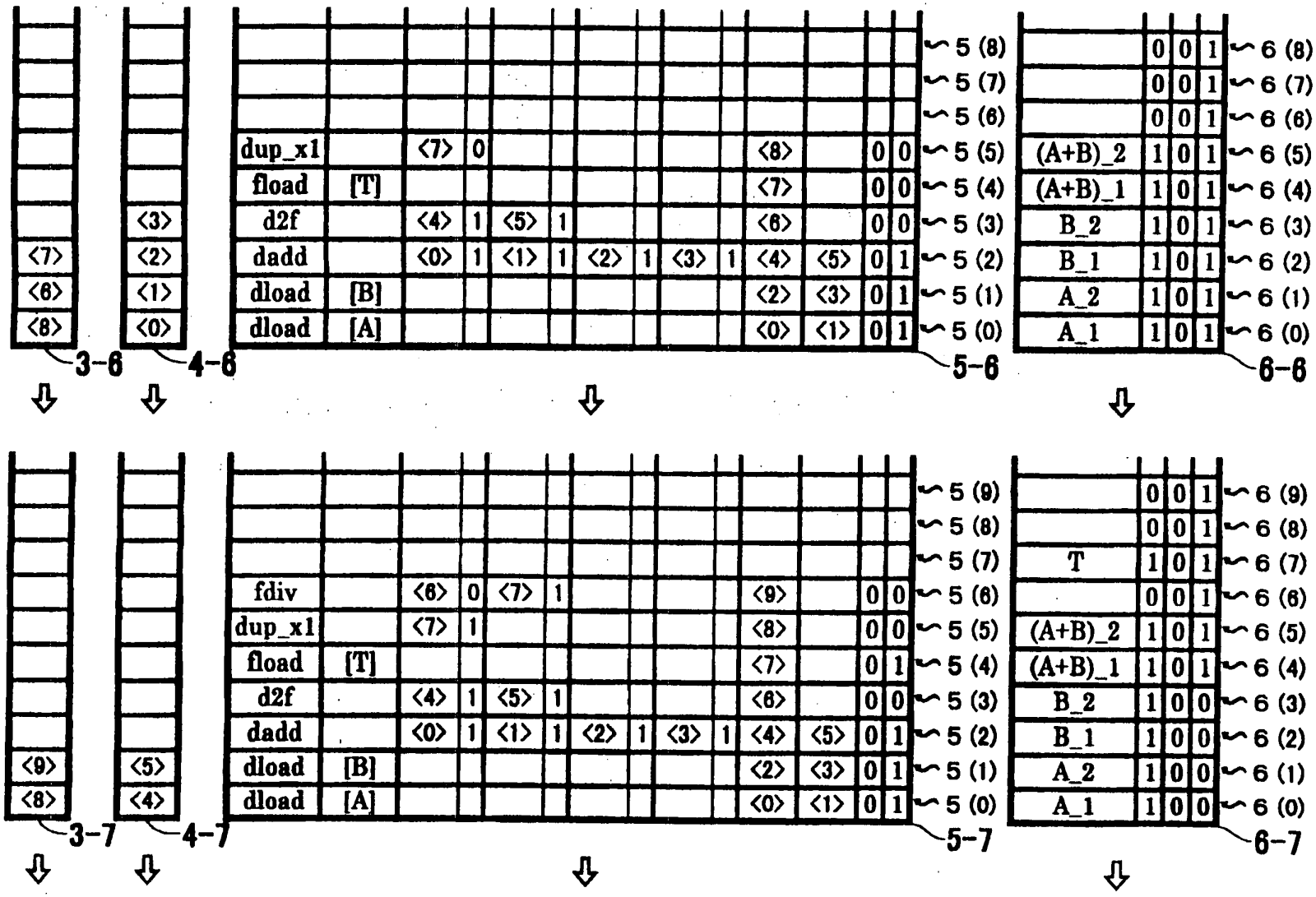
第 6 図



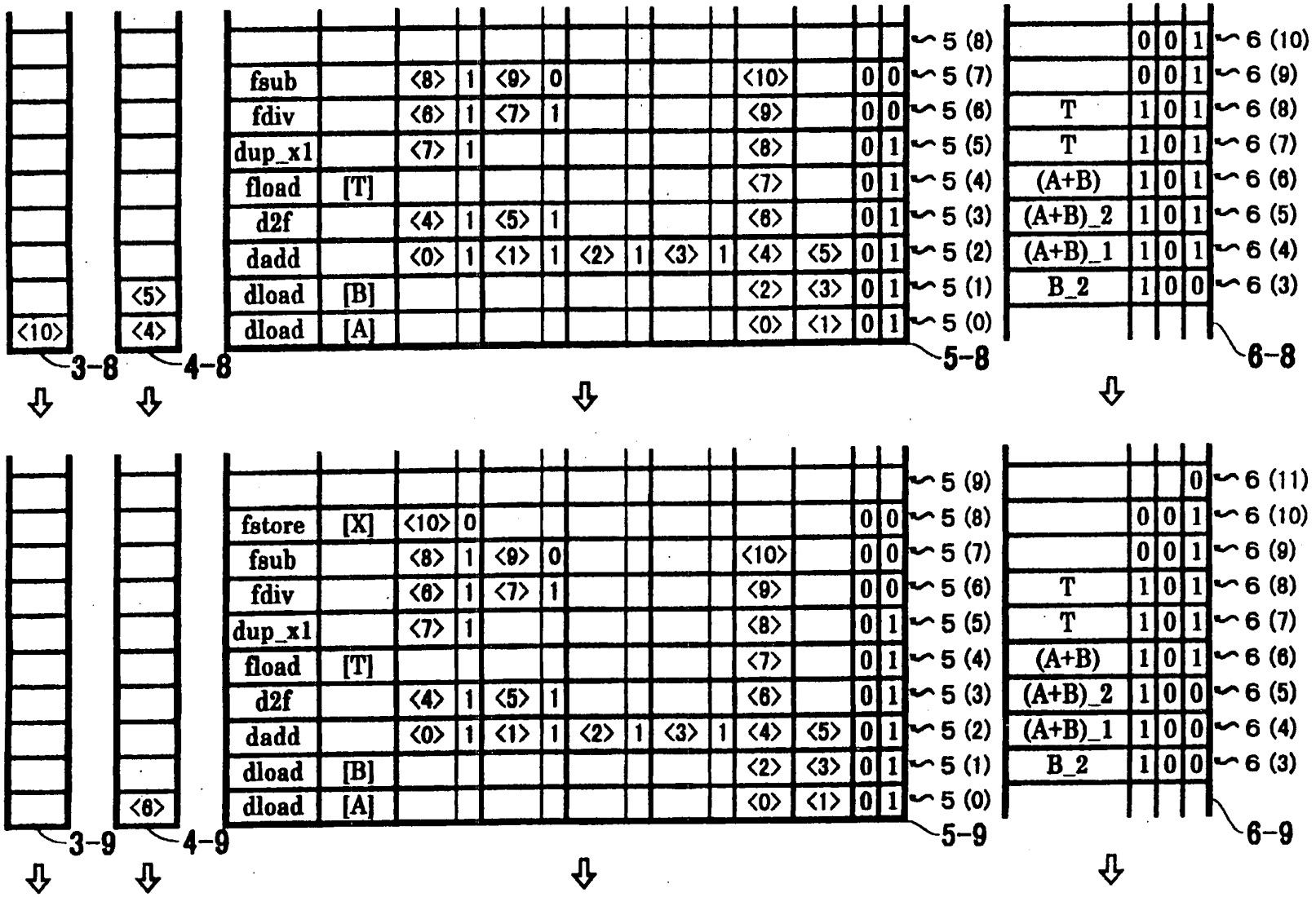
第 7 図


6 (7)
6 (6)
6 (5)
6 (4)
6 (3)
6 (2)
6 (1)
6 (0)
6-4
B_2
B_1
A_2
A_1
6 (8)
6-5
5 (7)
5 (6)
5 (5)
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-4
d2f
dadd
dload
dload
[B]
[A]
5-5
fload
[T]
4-4
4-5
3-4
3-5
<0>
<1>
<2>
<3>
<4>
<5>
<6>
<7>

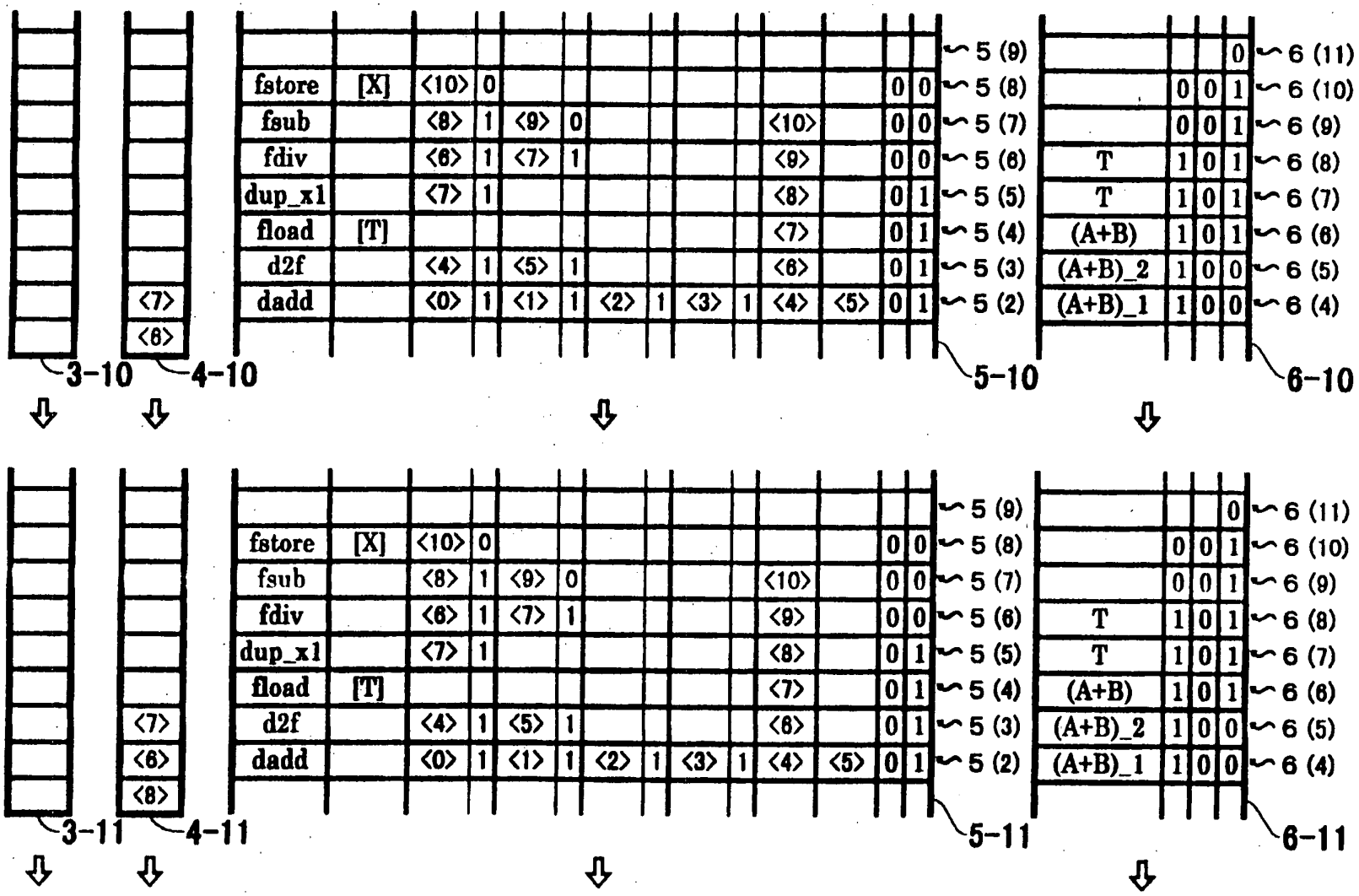
第 8 図



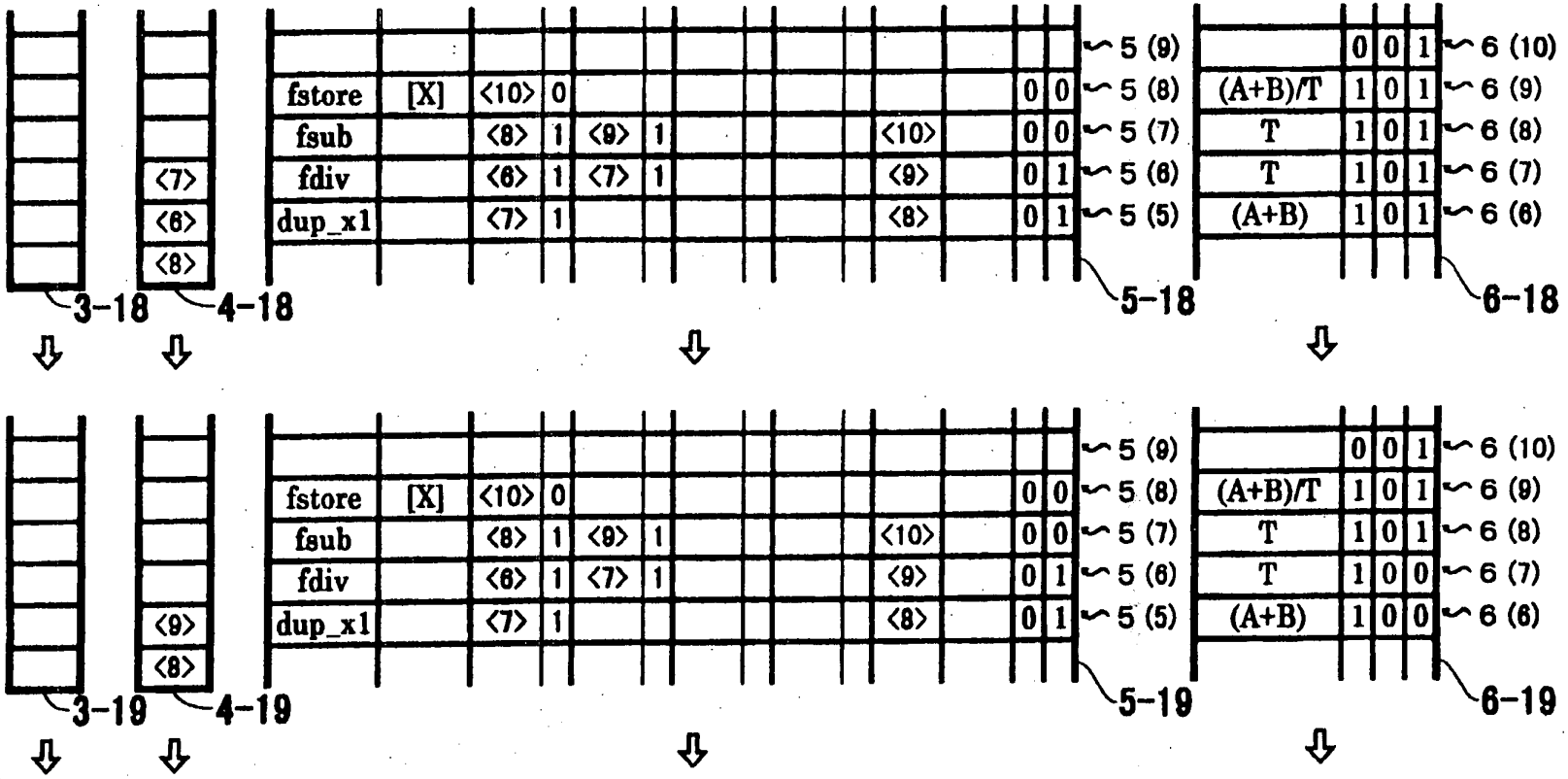
第 9 図


fsub
fdiv
dup_x1
fload
d2f
dadd
dload
dload
[T]
[B]
[A]
5 (8)
5 (7)
5 (6)
5 (5)
5 (4)
5 (3)
5 (2)
5 (1)
5 (0)
5-8
6 (10)
6 (9)
6 (8)
6 (7)
6 (6)
6 (5)
6 (4)
6 (3)
T
T
(A+B)
(A+B)_2
(A+B)_1
B_2
6-8
3-8
4-8
fstore
[X]
5 (9)
5 (8)
6 (11)
6 (10)
5-9
6-9
3-9
4-9

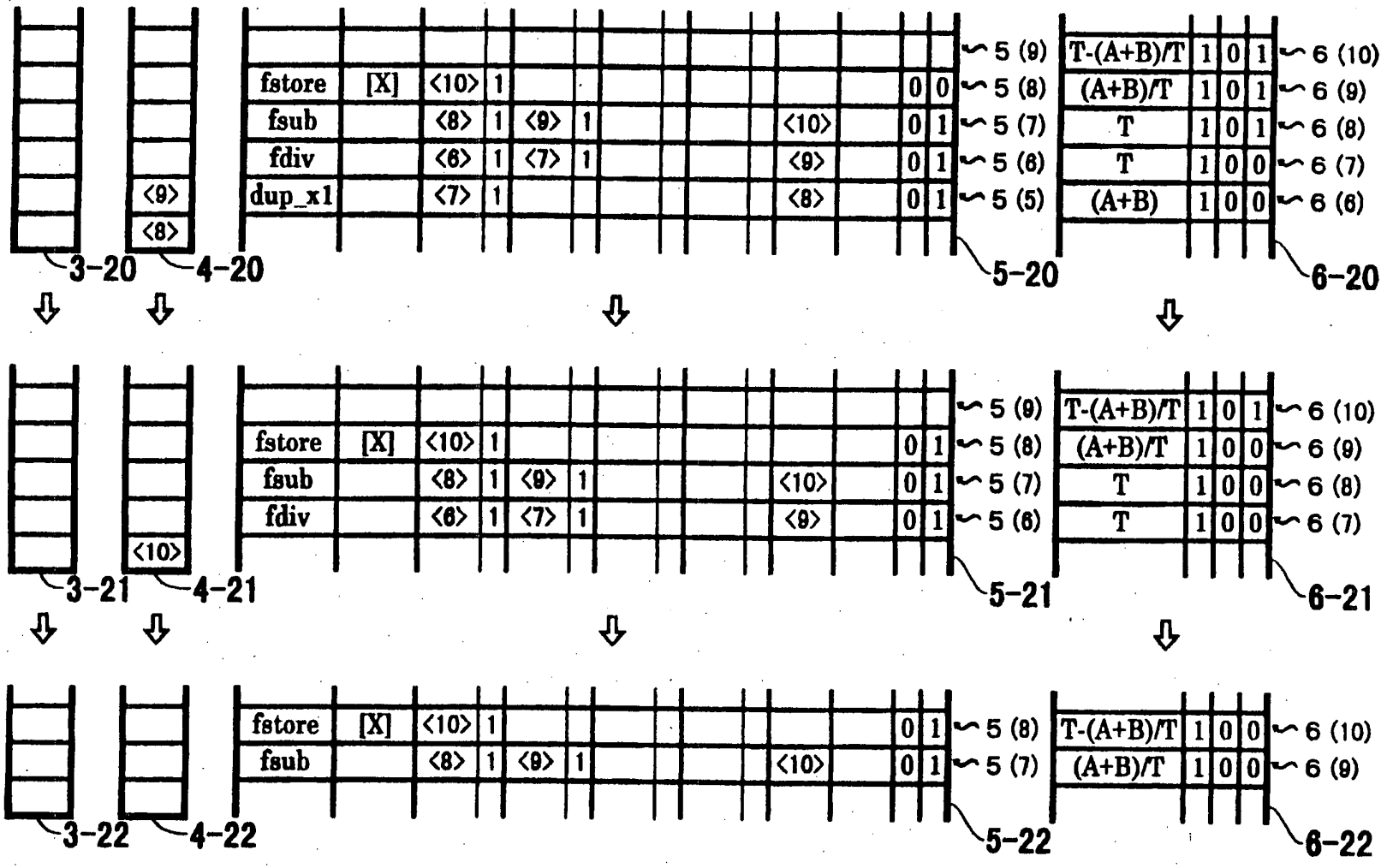
第 10 図



第 11 図



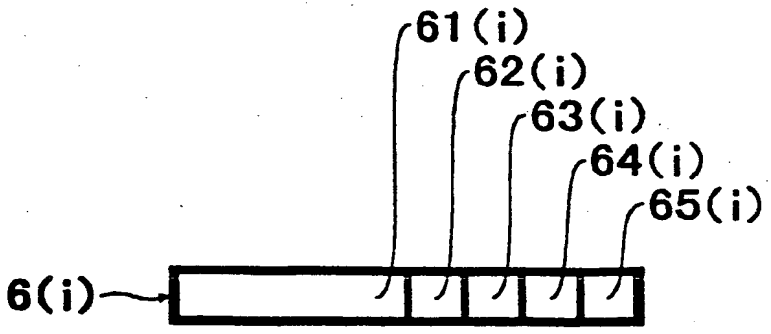
第 12 図



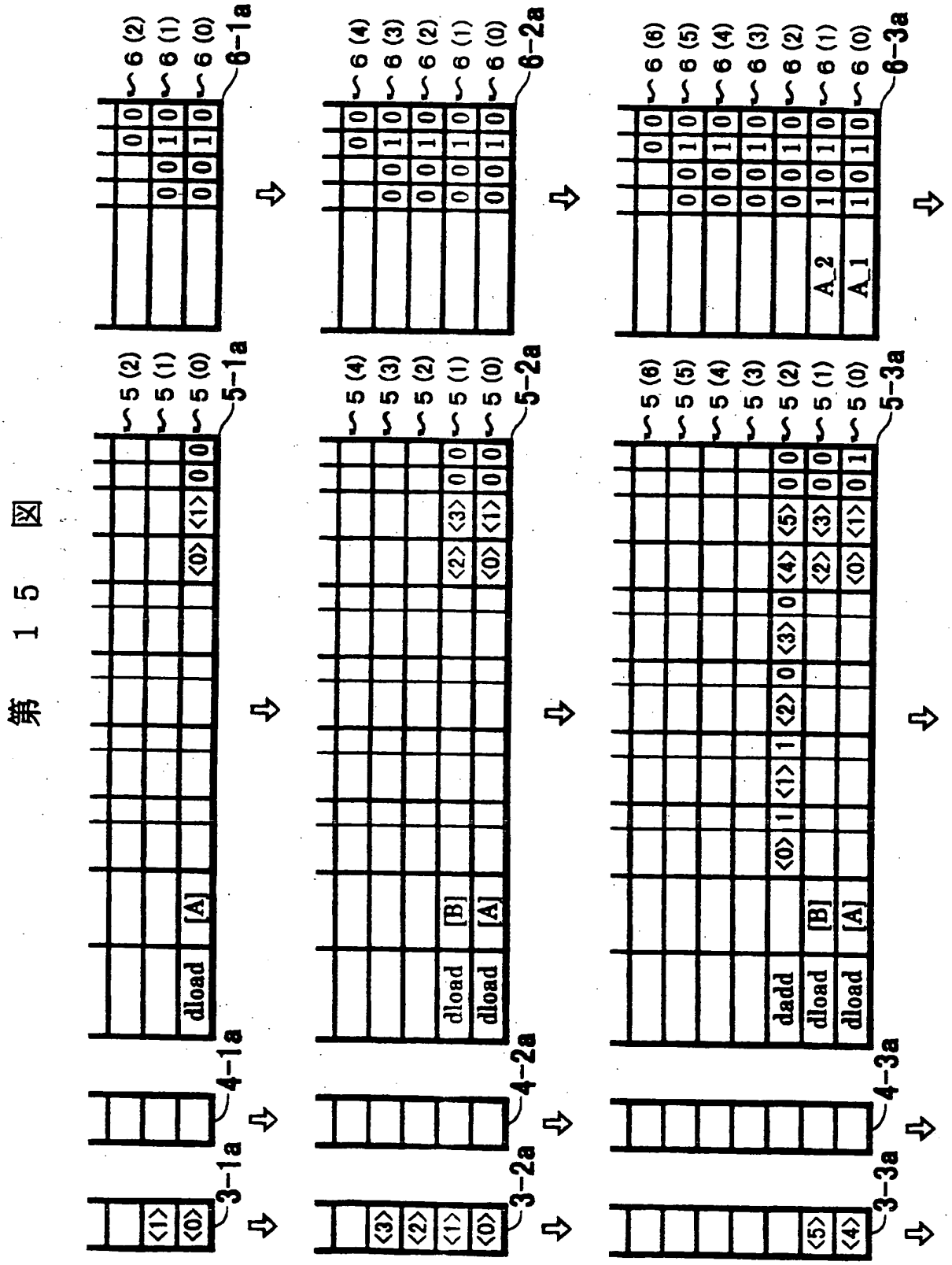
第 １３ 図

PP_OF_APS の増分	ＡＰＳの操作内容	ＩＱの設定内容
+2	NC NC NC f5 f6	dload [A] f1 f2 dload [B] f3 f4 dadd f1 f2 f3 f4 f5 f6
+1	NC NC f3 f1 f2	d2f s1 s0 f1 fload [T] f2 dup_x1 f2 f3
−3	NC NC NC NC NC	fdiv s1 s0 f1 fsub s2 f1 f2 fstore [X] f2

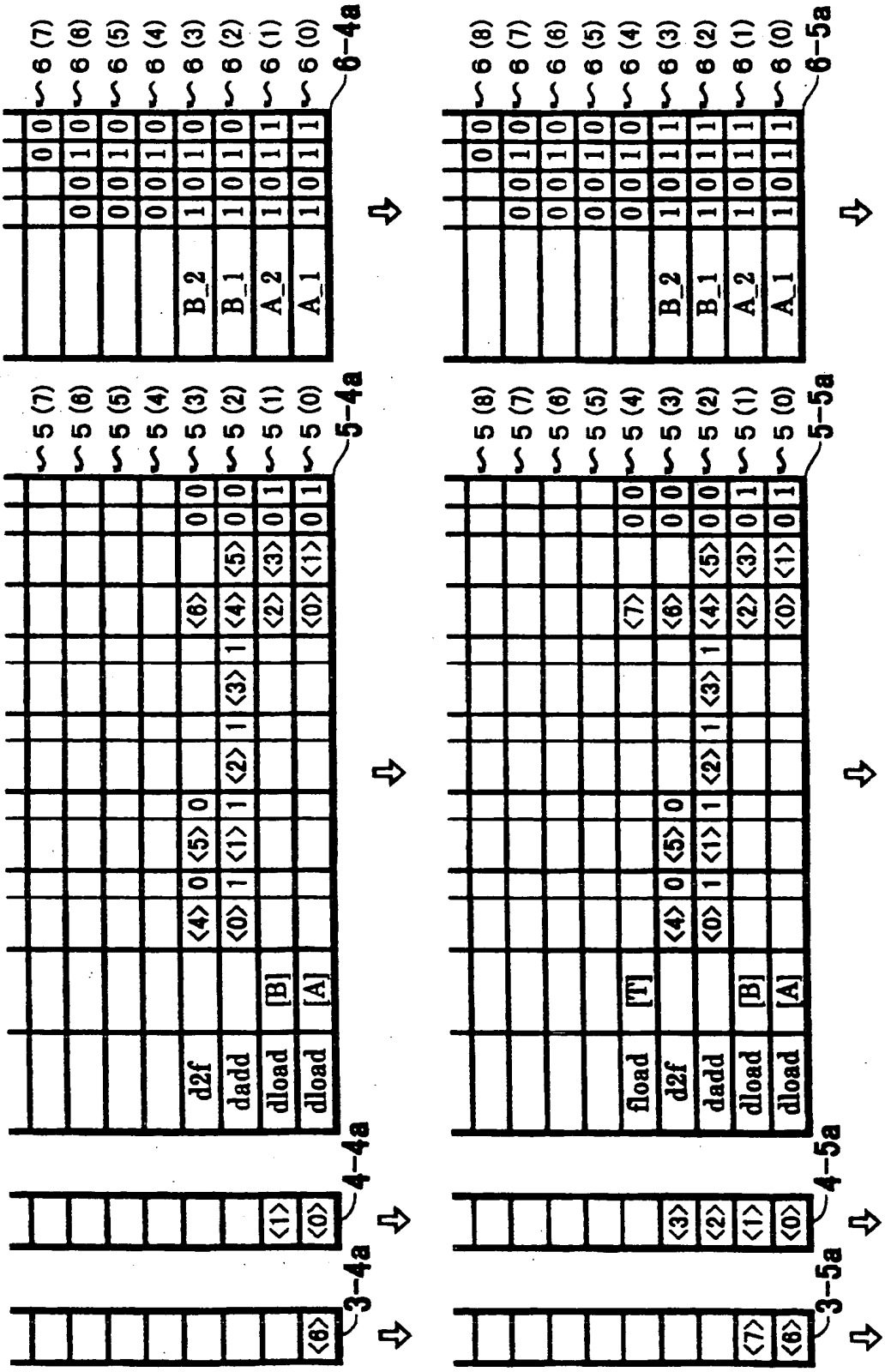
第 １４ 図



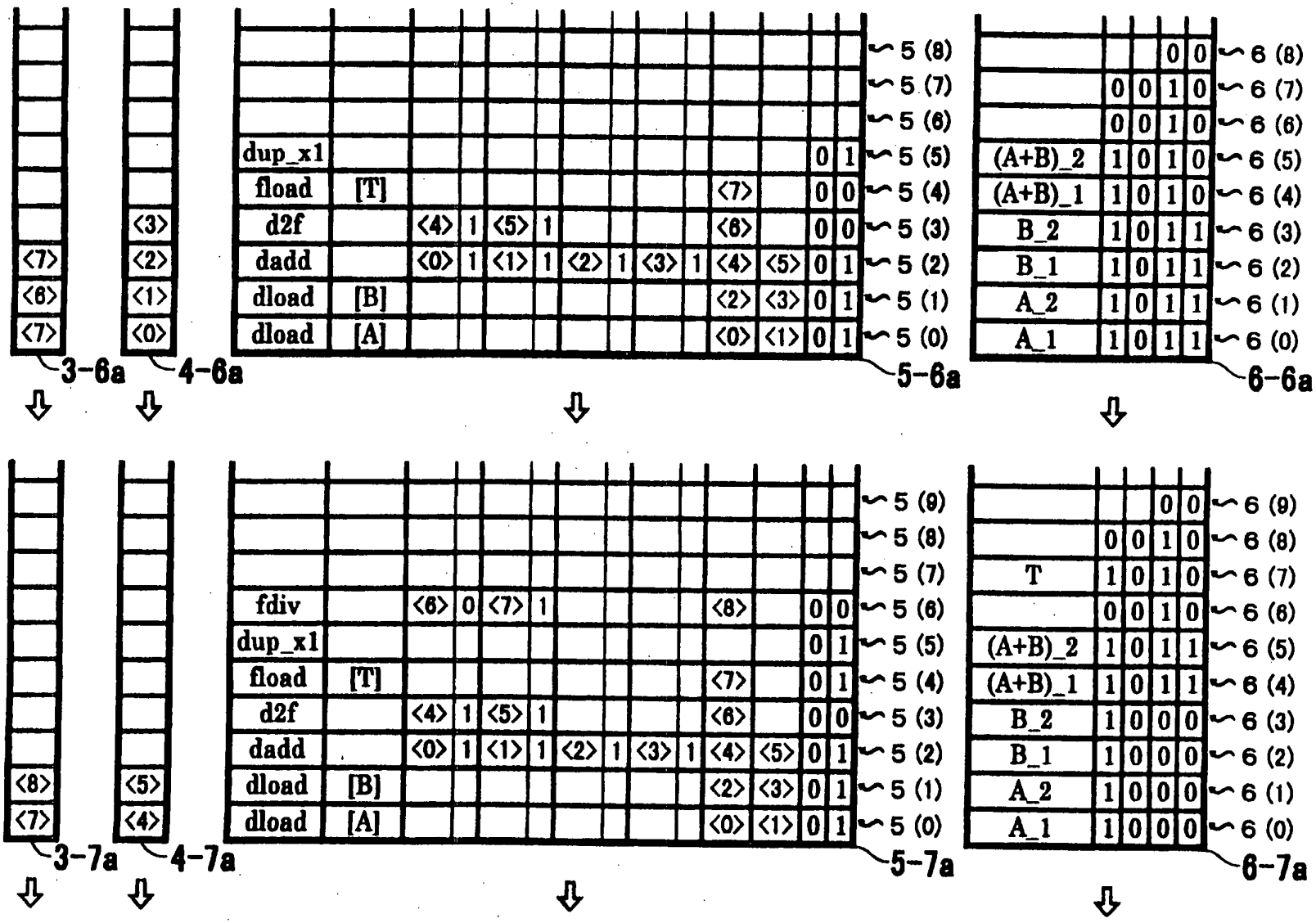
第 15 図



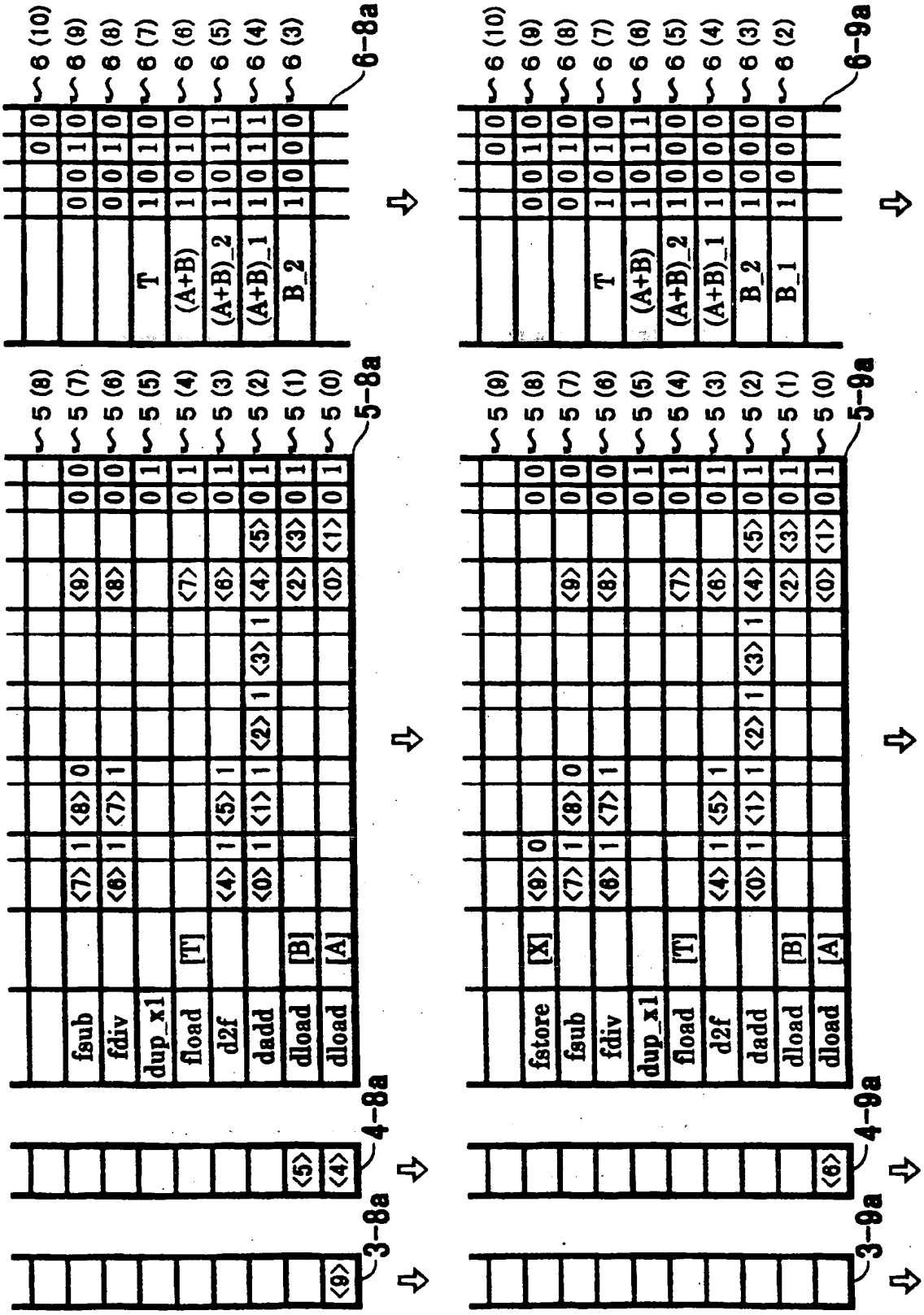
第 16 図



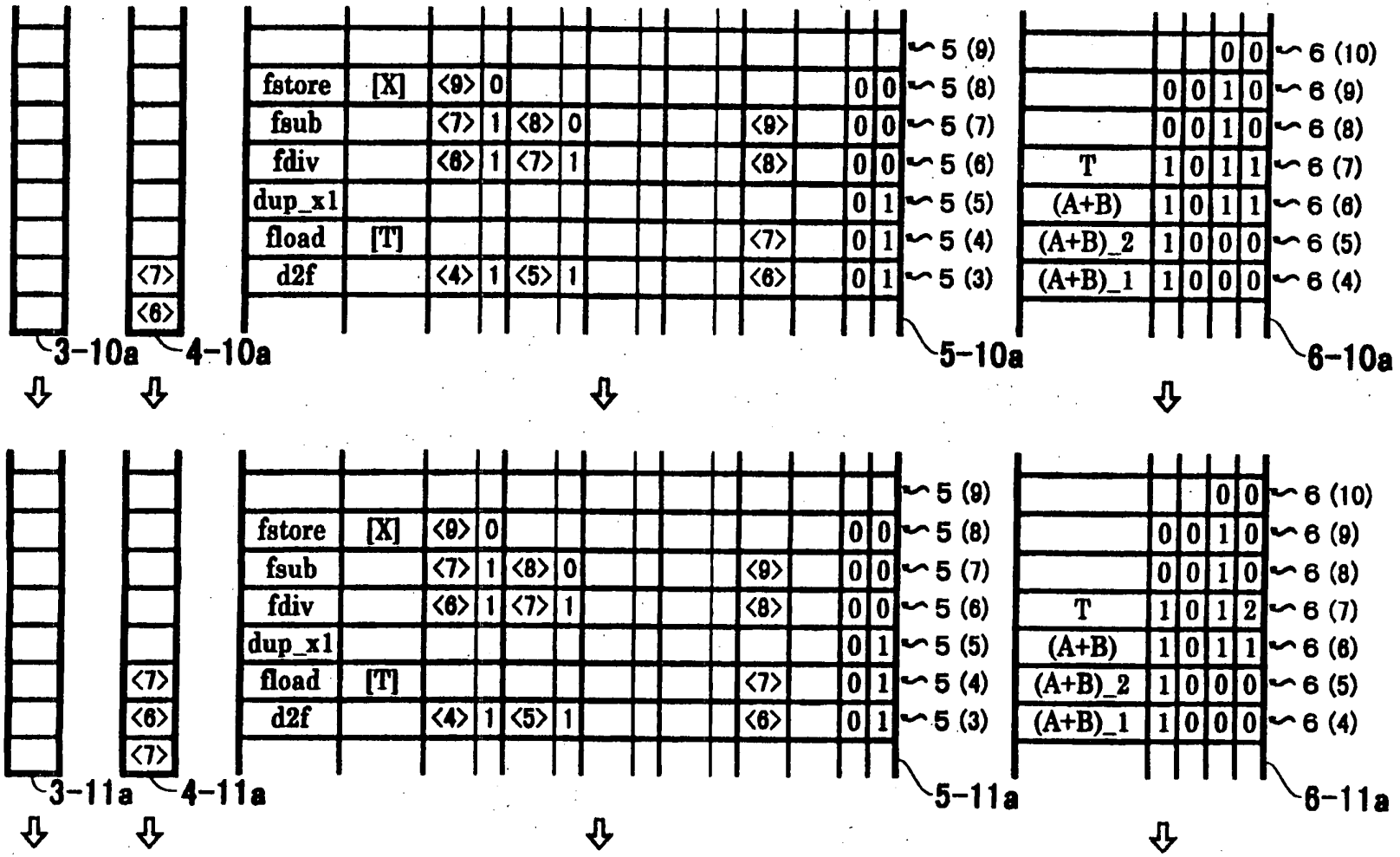
第 17 図



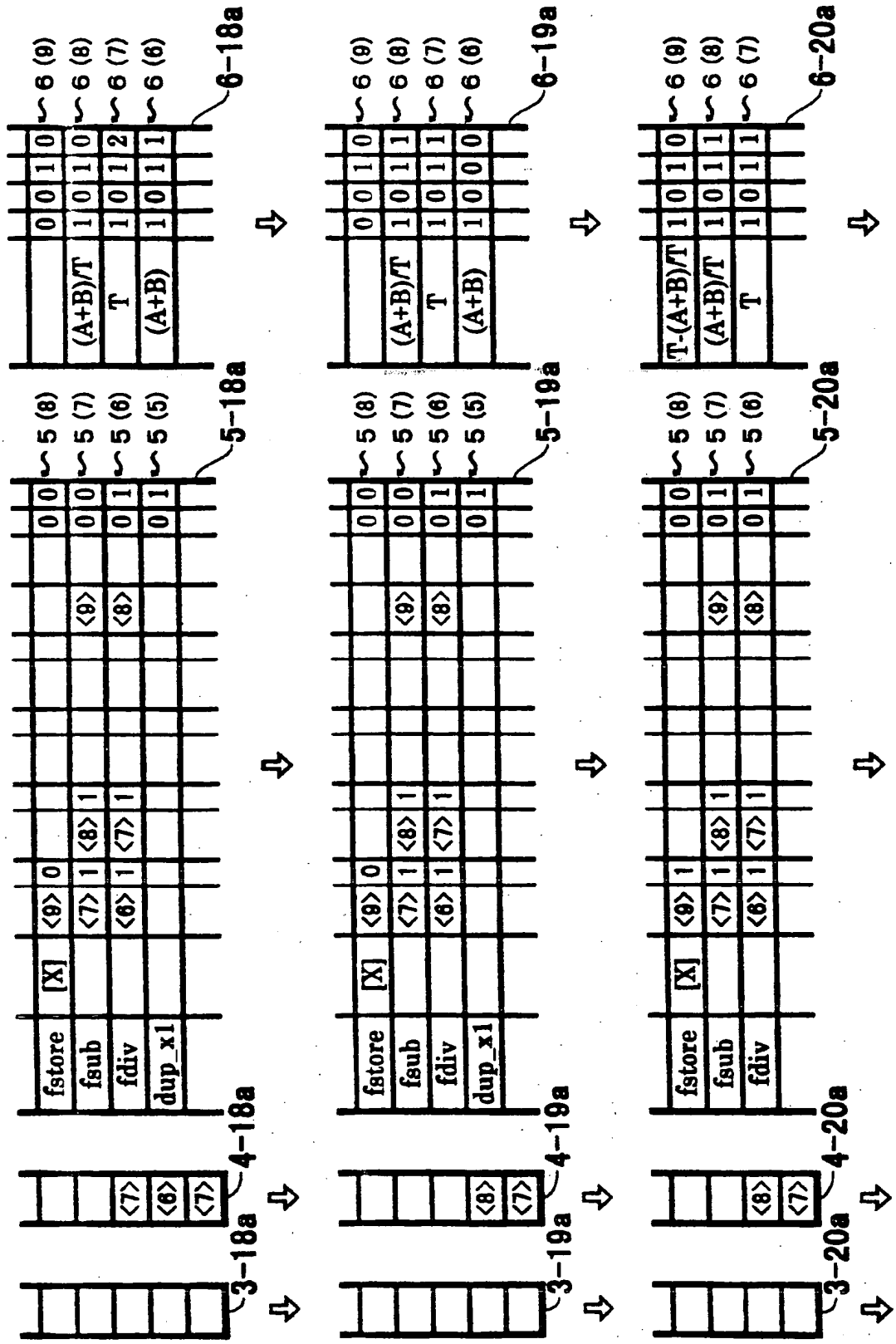
第 18 図



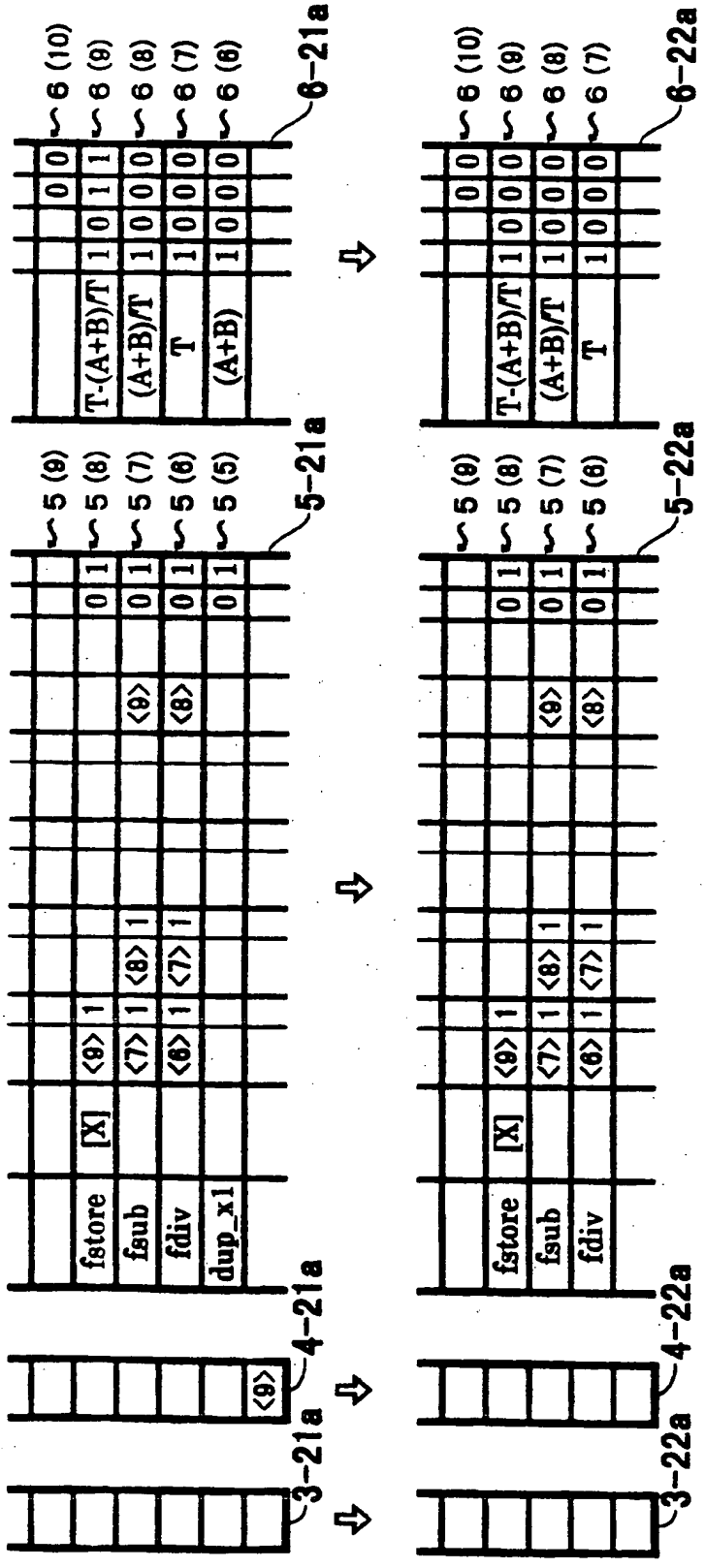
第 19 図



第 20 図



第 21 図



INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP98/05230

A. CLASSIFICATION OF SUBJECT MATTER
Int.Cl[6] G06F9/38, G06F15/82

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Int.Cl[6] G06F9/38, G06F15/82

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Jitsuyo Shinan Koho 1940-1999 Toroku Jitsuyo Shinan Koho 1994-1999
Kokai Jitsuyo Shinan Koho 1971-1995

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JP, 2-260082, A (Hajime Seki), 22 October, 1990 (22. 10. 90) (Family: none)	1-6
A	JP, 62-2330, A (NEC Corp.), 8 January, 1987 (08. 01. 87) (Family: none)	1-6

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 January, 1999 (21. 01. 99)	9 February, 1999 (09. 02. 99)
Name and mailing address of the ISA/ Japanese Patent Office	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1992)

国際調査報告

国際出願番号　PCT／JP98／05230

A.　発明の属する分野の分類（国際特許分類（IPC））
Int. Cl⁶ G06F 9／38
Int. Cl⁶ G06F15／82

B.　調査を行った分野

調査を行った最小限資料（国際特許分類（IPC））
Int. Cl⁶ G06F 9／38
Int. Cl⁶ G06F15／82

最小限資料以外の資料で調査を行った分野に含まれるもの
日本国実用新案公報　1940－1999年
日本国公開実用新案公報　1971－1995年
日本国登録実用新案公報　1994－1999年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

C.　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求の範囲の番号
A	JP，2－260082，A（関　一），22．10月．1990（22．10．90）（ファミリーなし）	1－6
A	JP，62－2330，A（日本電気株式会社），8．1月．1987（08．01．87）（ファミリーなし）	1－6

☐ C欄の続きにも文献が列挙されている。　☐ パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「A」特に関連のある文献ではなく、一般的技術水準を示すもの
「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「O」口頭による開示、使用、展示等に言及する文献
「P」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日	国際調査報告の発送日
21．01．99	09.02.99

国際調査機関の名称及びあて先	特許庁審査官（権限のある職員）		
日本国特許庁（ISA／JP） 郵便番号100－8915 東京都千代田区霞が関三丁目4番3号	中野　裕二	5B	9462
	電話番号　03－3581－1101　内線　3547		

様式PCT／ISA／210（第2ページ）（1998年7月）